VISUAL EXTEND 8.0

Die umfassende Software – Entwicklungsumgebung zur einfachen Applikationsentwicklung mit Microsoft Visual FoxPro!

Deutsches Benutzerhandbuch

dFPUG c/o ISYS GmbH

INHALTSVERZEICHNIS

1. Einleitung

von Rainer Becker

Woher kamen wir, wo stehen wir jetzt und wo wollen wir eigentlich hin? Diese Fragestellung ist vielleicht etwas unüblich in einer Einleitung für ein Handbuch eines Frameworks - aber dafür ist es hoffentlich dennoch lesenswert und nebenbei vielleicht unterhaltsam und informativ. Und sofern zumindest eine gewisse Übereinstimmung der Meinungen in den nachfolgenden Abschnitten festgestellt werden kann, sollten wir in Zukunft recht gut miteinander auskommen, weil Sie dann besser wissen, wo wir hin wollen und in wie weit unsere Zielrichtung mit Ihren Vorstellungen übereinstimmt und wiederum Sie in Ihren Zielen unterstützt.

1.1. Die Vorteile von FoxPro

Sie kennen den Spruch vielleicht: 3x umgezogen ist wie 1x abgebrannt! Wer sich von FoxPro/DOS nach FoxPro/Windows durchgekämpft hatte, war mit diesem ersten Umzug zwar schwer beschäftigt, aber da das Grundkonzept schon vorher recht modern war, ließ sich dieser Umstieg meist dennoch recht gut bewältigen. Und was bekam ein FoxPro-Entwickler mit seiner Entwicklungsumgebung - ungefähr folgendes auch heute immer noch sehr attraktiv klingendes Angebot:

- Keine Laufzeitgebühren für erstellte Anwendungen
- Keine Lizenzkosten für die integrierte mitlieferbare Datenbank-Engine
- Keine Lizenzkosten für den integrierten mitlieferbaren Berichtsgenerator
- Integrierte Werkzeuge für Masken-, Menü- und Berichtserstellung
- Integrierte Werkzeuge für Hilfeanbindung, Debugging und Installation
- Assistenten aller Arten für die Automatisierung verschiedener Entwicklungsaufgaben (Builders und Wizards)
- Leistungsfähigere Programmiersprache als die derzeit mal wieder beliebten Script-Sprachen
- Leistungsfähige relationale Datenbank mit alternativer Client/Server-Backend-Datenbank (meist MSDE/SQL-Server)
- Hybrider Datenbankzugriff sowohl mit satzorientiertem als auch SQL-basiertem Zugriff

- Kalkulierbarer Erstellungsaufwand von datenorientierten Anwendungen für alle Einsatzbereiche
- Performante Anwendungen auf der schnellsten verfügbaren PC-Datenbank

Der Umzug von FoxPro/DOS nach FoxPro/Windows war im Vergleich zu dem Umzug von FoxPro/Windows nach Visual FoxPro im Nachhinein betrachtet dann doch eher einfach, da dieser zweite Umzug sich aufgrund völlig neuer Konzepte wie Objektorientierung und Vererbung sowie Methoden, Ereignissen und Eigenschaften wesentlich herausfordernder gestaltete. So einige Entwickler haben sich bis heute wider Erwarten und leider erfolgreich vor diesem Umstieg gedrückt - fragt sich nur, wie lange das noch gut gehen soll. Insbesondere, da dadurch auch die KnowHow-Basis und Entscheidungsfähigkeit fehlt, möglicherweise anderswohin zu wechseln.

Die im ersten Abschnitt genannten Vorteile von FoxPro wurden in Visual FoxPro beibehalten, dazu kamen noch folgende wesentlichen Erweiterungen:

- Vollständig objektorientierte Entwicklungsumgebung mit einer Vielzahl von leistungsfähigen Klassen, Vererbung, Container-Konzept und entsprechenden neuen oder erweiterten Werkzeugen
- Hybrider Programmieransatz sowohl prozedural als auch objektorientiert - ähnlich der alternativen Datenzugriffsmöglichkeit auf satzorientierter oder mengenorientierter (SQL) Basis
- Unterstützung aller modernen Technologien wie ActiveX, OLE-Automation - und mittlerweile natürlich WebServices, XML und COM
- Schnelle Webseitengestaltung durch Datenbankintegration und Stringfunktionen - z.B. mit Active FoxPro Pages oder WebConnect

Genial war nicht nur die Beibehaltung des bisherigen hybriden Ansatzes beim Datenzugriff (satz- und mengenorientiert) sondern das nunmehr ebenfalls hybride Programmiermodell (prozedural und OOP). Beliebige Wechsel zwischen SKIP und SQL waren nunmehr genauso möglich wie der Aufruf einer Prozedur aus einer Objekt-Methode. Gnadenlos praktisch ist dabei die direkte Editierbarkeit der in Tabellen gehaltenen Klassen- und Maskendefinitionen (sog. Metadaten), was von jeher das Umbiegen der Vererbungshierarchie erlaubte zwecks nachträglicher Korrektur oder Einfügung von Zwischenebenen. Als ebenso praxisnah erwies sich das hierarchische Container-Konzept statt der ansonsten von Microsoft verfolgten SimpleFrame-Oberfläche, die keine zwei gleichnamigen Objekte im gleichen Formular erlaubt und damit die eigentlich interessante Wiederverwendung von aggregrierten Objekten zur Qual macht.

Wessen Spezialität es natürlich vorher schon war, Code zu kopieren und anzupassen, statt eine erweiterte Standardfunktion zu bauen um wenig gesparte Entwicklungszeit gegen viel Wartungsaufwand zu tauschen, konnte das in der neuen Welt natürlich weiterhin tun und damit jedweden theoretischen Nutzen von vorneherein bzw. weiterhin über den Haufen schießen, wie oft genug geschehen. Umgekehrt führt der verzweifelte Versuch, wirklich alles in Klassen abzubilden und für jede denkbare Ebene eine eigene Schicht leerer Zwischenklassen einzuführen natürlich zu einem viel zu komplexen Entwicklungskonzept, welches neuen Projektmitarbeitern überhaupt nicht mehr vermittelbar ist. Wie immer liegt wohl auch hier die Wahrheit in der Mitte.

Wirklich rund wurde die Entwicklung aber erst mit dem großen Sprung des Funktionsumfangs bei der stabilen und leistungsfähigen Version Visual FoxPro 6.0, die wohl die am häufigsten eingesetzte Version von Visual FoxPro ist. Aufgrund der Abwärtskompatibilität war jedes weitere Update beginnend mit Visual FoxPro 3.0 eher unproblematisch, so dass man jederzeit auf die Folgeversion umsteigen konnte, auch wenn die Version 7.0 in der deutschen Variante wohl einige unerfreuliche Speicherlecks enthielt.

1.2. Noch mehr Vorteile mit Visual FoxPro 8.0

Die aktuelle Version Visual FoxPro 8.0 ist eine wesentliche Erweiterung der Vorgängerversion mit einem ebensolchen Sprung in der Entwicklung wie bei der beliebten Version 6.0. Das wird besonders deutlich in der entsprechenden vergleichenden Darstellung von Christian Desbourse mit einer quantitativen Gegenüberstellung aller bisherigen Versionen von Visual FoxPro - erschienen in der Ausgabe 12.0 der Loseblattsammlung FoxX Professional aber auch online im dFPUG-Portal sowie auf seiner Homepage verfügbar. Leider ist dieser große Schritt in der Entwicklung noch nicht bekannt genug und wird von vielen FoxPro-Entwicklern noch nicht wirklich gesehen geschweige denn von dem daraus resultierenden Nutzen für die praktische Arbeit.

Die wichtigsten Features der Version Visual FoxPro 8.0 sind:

- Neue Werkzeuge wie Taskpane, Toolbox und Code Referenz-Suche
- Verbesserungen bei Werkzeugen wie Berichtsgenerator, Menüdesigner, View-Designer und weiteren
- Neue Builder wie Dataenvironment-Builder und XML-Webservice-Builder
- Datenbankerweiterung um Auto-Increment-Felder, Ausdrücke und SQL-Funktionalität

- Völlig neue flexible Fehlerbehandlung mit dem TRY-CATCH-Konstrukt
- Umfangreiches Eventbinding innerhalb und außerhalb der eigenen Anwendung
- Endlich eigene visuelle Subklassen für Page, Column, Header, Option-Button, Commandbutton
- Neue Klassen wie Collection, XML-Adapter, CursorAdapter oder auch nur Empty.
- Unterstützung von Hyperlinks, verzögertes Databinding usw. in Objekten
- Weitere Verbesserungen im Bereich COM-Server, Intellisense - eigentlich fast überall

Zu Recht wird es als das umfangreichste Update seit Visual FoxPro 6.0 bezeichnet. Viel zu lernen gibt es also für die Anwender, insbesondere für diejenigen unter uns, die sich ein oder mehrere Updates zwischendrin gespart haben. Sprich die Mehrheit, die jetzt von Visual FoxPro 6.0 oder gar 5.0 direkt auf Visual FoxPro 8.0 umsteigen und dann die Erweiterungen von 2-3 Versionen auf einmal lernen dürfen. Mitleid ist dabei aber eigentlich nicht angebracht, denn jeder Söldner modernisiert seine Ausrüstung und schärft sein Messer - nur FoxPro-Entwickler lassen gerne mal 1-2 Updates und damit 3-5 Jahre Weiterentwicklung in einem Markt aus, wo 5 Jahre eine Generation darstellen. Es ist natürlich richtig: updaten kostet nicht nur die Updategebühren sondern auch eine Menge Zeit für die Einarbeitung - und warum sollte man eine bestehende Anwendung updaten, wenn der direkte Nutzen für den zahlenden Anwender nicht erkennbar ist. Aber es ist auch richtig: Lieber schrittweise mitlernen als jahrelang nicht updaten und dann einen riesengroßen Schritt bewältigen müssen, für den man im Arbeitsalltag dann oft doch keine Zeit hat und daran dann scheitert.

Aber fast genauso wichtig an der neuen Version wie die Vielzahl der praxisnahen technischen Erweiterungen sind auch die Ankündigungen von Microsoft in diesem Zusammenhang, zum Beispiel:

- Wartungszusage von Microsoft für Visual FoxPro 8.0 bis 2010 und damit länger als für jedes andere Produkt - mit vermuteter Verlängerung im Folgejahr für die nächste Version Visual FoxPro 9.0
- Verfügbarkeit eines Service Pack 1 für Visual FoxPro 8.0 - der Standardgrund in Deutschland "Ich warte erst auf das erste Service Pack da eine .0-Version von Microsoft sowieso nix taugt" ist damit ausgehebelt

- Verfügbarkeit eines aktualisierten OLE-DB-Providers für Visual FoxPro 8.0

Ein Wermutstropfen ist natürlich die generelle Einstellung der Lokalisierung von Benutzeroberfläche und Hilfe für Deutsch und Spanisch (alle anderen Sprachunterstützungen wurden schon mit der Version 6.0 bzw. spätestens mit der Version 7.0 eingestellt, was hierzulande aber sowieso keiner gemerkt hat). Deshalb helfen folgende Informationen weiter:

- Kostenlose Verfügbarkeit einer deutschen Benutzeroberfläche im dFPUG-Dokumenten-Portal
- Verfügbarkeit eines deutschen Updatebuchs zu VFP 8.0 über den dFPUG-Bestellservice
- Ankündigung einer deutschen Dokumentation / Hilfedatei seitens der dFPUG

In Kombination der deutschen Benutzeroberfläche mit der Hilfedatei von Visual FoxPro 7.0 (wenn man diese denn besitzt) und dem Updatebuch zur Version 8.0 liegt eigentlich alles (bis auf die Assistenten) in Deutsch vor, so dass der seitens Microsoft natürlich Kosten sparende Verzicht auf die Lokalisierung sich nur wenig auf den Arbeitsalltag auswirken sollte und mit der Verfügbarkeit der deutschen Dokumentation entfällt dieser Punkt dann sowieso.

1.3. Die Vorteile von Visual Extend

Das Framework Visual Extend hat übrigens diesen Ursprung von FoxPro von vorneherein nicht verlassen und das war unseres Erachtens ein wichtiger Grund für den Erfolg des Produktes. Die zusammenfassenden Marketing-Schlagworte für Visual Extend sind seit vielen Versionen unverändert und zutreffend die folgenden:

- Erstellen Sie Ihre eigenen Office Compatible Anwendungen!
- Beinhaltet mächtige Builder für Forms, Grids, Picklists sowie anspruchsvolle One-to-many-Formulare!
- Erstellen Sie mit dem Visual Extend Application Wizard neue Projekte in der Sprache Ihrer Wahl!
- Fügt sich optimal in Ihre bestehende Visual FoxPro Umgebung ein!
- Lassen Sie die Visual Extend Builder für Sie die harte Programmierarbeit erledigen!
- Mit Visual Extend werden auch Sie zum Visual FoxPro Profi!

Die bestehenden Versionen von Visual Extend unterstützen den Entwickler unter Visual FoxPro mit folgenden Werkzeugen:

- Automatisierte Projektvorbereitung
- Office-kompatible Standardmenüs mit Favoriten und Symbolleiste
- Benutzerverwaltung und Zugriffsrechte
- Datenbankwartung, Fehlerprotokoll, Systemsperren
- Anpassbarer Optionen- und Info-Dialog
- Formular-Builder mit Reiterdefinition für Bearbeitung und Listendarstellung
- Grid-Builder mit inkrementeller Suche und Speichern von Benutzereinstellungen
- Die gleiche Builder-Kombination für 1:n-Formulare
- Builder für verschiedenen Arten von Auswahllisten
- Werkzeuge für Sprachverwaltung und Nachrichtentexte
- Integration von Berichts- und Filterfunktionen
- Standardklassen für ActiveDesktop, Moverboxen, Fortschrittsanzeige, Datumsauswahl usw.
- Standardfunktionen für Primärschlüssel, Protokollierung
- Mandantenfähigkeit und Client/Server-Unterstützung
- Hilfe-Generierung und OLE-Unterstützung

Damit werden sämtliche Standardbereiche einer Applikationsentwicklung rein unter Visual FoxPro oder als 2-Schichten-Anwendung nach dem Client/Server-Prinzip mit einer Backenddatenbank (wie z.B. SQL-Server oder MSDE) erfolgreich abgedeckt. Nicht weiter verfolgt wurden die Bemühungen, eine vollständige 3-Schichtenarchitektur mit separierter Geschäftslogik zu implementieren, da der Aufwand für diese Architektur und das notwendige theoretische Hintergrundwissen für eine preiswerte Erstellung von ansprechenden Anwendungen im Bereich der KMUs meist viel zu hoch ist.

Neben diesen technischen Gründen sprechen aber auch einige weitere Argumente ganz gewichtig für Visual Extend:

- Sehr lange Erprobungs- und Reifephase von Visual FoxPro/Visual Extend 3.0 über die Versionen 5.0, 6.0, 7.0 und 7.1 zur heutigen Version Visual FoxPro / Visual Extend 8.0

- Grosse Anwenderbasis, mit jeweils vielen hundert Anwendern - zu gleichen Teilen im deutschsprachigen wie im internationalen Raum und damit unabhängiger von lokalen Entwicklungen
- Stabiler Hersteller mit langjähriger Erfahrung in Visual FoxPro und treuen Kunden (vorher Devigus Engineering, heutzutage die dFPUG selbst)
- Umfangreiches Online-Angebot mit Forum / Newsgroups und einer Vielzahl von Nachrichten, Dokumenten, Vorträgen, Artikeln und Slideshows rund um das Produkt zu fast jeder Fragestellung

Allein schon aus den hier genannten Gründen werden z.B. mehr Lizenzen von Visual Extend im deutschsprachigen Raum verkauft als von allen anderen Framework-Anbietern, was wiederum die Stabilität der Weiterentwicklung und Wartung des Produktes sehr positiv beeinflusst, auch wenn natürlich Visual Extend genauso wie Visual FoxPro selbst unter der meist geringen Bereitschaft der Anwender zum Erwerb eines Updates leidet.

1.4. Noch mehr Vorteile mit Visual Extend 8.0

Mit der aktuellen Version von Visual Extend wird die gleiche Richtung fortgeführt, die auch die Vorversionen sowie Visual FoxPro so beliebt gemacht haben mit Konzentration auf schrittweise Weiterentwicklung unter Beibehaltung einer weitestgehenden Abwärtskompatibilität. Die neue Version Visual Extend 8.0 bietet dabei ein so umfangreiches Update wie noch nie. Da die reine Ablauffähigkeit unter der neuen Version Visual FoxPro 8.0 mit dem kostenlosen Zwischenrelease Visual Extend 7.1 bereits abgedeckt werden konnte, wurde der Schwerpunkt ohne die sonst notwendige zeitliche Einschränkung wegen möglichst schneller Lieferung zur Ablauffähigkeit unter der neuesten Version von Visual FoxPro auf die Runderneuerung und wesentliche Erweiterung in einer Vielzahl von Bereichen gesetzt. Hier eine grobe Übersicht über die wichtigsten der neuen Features:

- integrierter eigener Menü-Designer
- Berichtsausgabe in PDF-Dateien
- E-Mail-Versand aller Berichtsausgaben
- Öffnen-Dialog im Windows XP-Style
- Unterstützung des Treeview Controls
- Builder für cTreeViewForm und cTreeViewOneToManyForm
- Entwicklerseitig konfigurierbare Produktaktivierung für 32 Module
- Konfigurierbare Skripte für Internetdownload incl. Anwendungsupdate

- SQL-Server-Datenbankupdate für Clients
- Updateseite für cPickTextBox-Builder wie cPickFieldBuilder
- neuer tabellenbasierter cPickAlternate-Builder
- Application Manager als VFP8-TaskPane
- Verwendung von DataEnvironment-Klassen
- weitere Beispielanwendungen im Quellcode
- Verbesserungen für Grid- und Formbuilder
- erweiterter cSearchDialog mit 5+ Kriterien
- Integration einer Backup-Funktion
- neues XP-Layout des Login-Dialogs
- stark erweiterte OLE-Ansteuerung von Word
- cPickDate mit zusätzlichen Hotkeys
- neue Klasse cDatetime zur Eingabe von Datetime-Werten

Bei den umfangreichen Umfragen der dFPUG bezüglich der Anforderungen an die neue Version Visual FoxPro 8.0 stand der Berichtsdesigner immer unangefochten an erster Stelle. Dieser wird mit der neuen Version von Visual Extend um PDF-Erzeugung und eMail-Versand erweitert, welches den Nutzen bestehender Anwendungen ohne Mehraufwand erhöht. An zweiter Stelle stand der Menüdesigner, welcher ebenfalls in Visual Extend in der neuen Version umfangreich und elegant adressiert wird.

Natürlich musste auch etwas im Bereich Look & Feel unternommen werden. Denn nicht nur Treeviews sind zeitgemäß und Windows XP sieht schon schick aus, sondern erweiterte Picklisten und viele weitere Erweiterungen in der Benutzeroberfläche sind natürlich auch zwingend notwendig für die Erstellung leicht bedienbarer Anwendungen. Auf jeden Fall auch hier eine Vielzahl von Angeboten, mit denen man nicht nur neue Anwendungen schneller entwickeln kann, sondern auch bestehende Anwendungen leicht erweitern kann, um seinen eigenen Kunden eine neue überarbeitete Version anbieten zu können.

Da fast die Hälfte der größeren Entwickler auch im Bereich Client/Server tätig ist, wurde das Feature Datenbankupdate für diesen Bereich erweitert. Für die anderen kam eine Funktion zur Datensicherung hinzu, die sich leicht in die eigene Anwendung integrieren lässt und ohne weitere Lizenzkosten oder Installationsaufwand ZIP-Dateien erzeugt.

Und für diejenigen, die nicht nur Inhouse entwickeln, sondern Ihre Anwendungen auch auf dem freien Markt anbieten, haben wir endlich eine vernünftige Lösung nicht nur für das Update der Anwendung über das Internet sondern auch für die Produktaktivierung mit Einzelrechten für bis zu 32 Module

über einen variabel definierbaren Aktivierungsschlüssel. Das Verfahren hat sich bei Visual Extend selbst bewährt und wird deshalb in erweiterter Fassung allen Entwicklern bereitgestellt.

Gleichermaßen hat sich auch die Mehrsprachigkeit von Visual Extend bewährt, was man an der internationalen Verteilung der Anwender sehen kann. Auch dies wird deshalb in erweiterter Fassung allen Entwicklern bereitgestellt in Form zusätzlicher Sprachunterstützung der generierten Anwendungen von bisher Deutsch, Englisch, Französisch, Italienisch und Spanisch um Griechisch, Bulgarisch, Tschechisch, Niederländisch, Portugiesisch und Russisch mit nunmehr insgesamt 11 unterstützen Sprachen in der generierten Anwendung.

1.5. Die nächsten Versionen

Für die nächste Version Visual FoxPro 9.0 wurde für das zweite Halbjahr 2004 schon eine erste Liste von neuen Funktionen bekanntgegeben, unter anderem:

- Erweiterbarer Berichtsdesigner mit Ereignissen und Rechten
- Erweiterungsmöglichkeit von Berichten zur Laufzeit
- Aufhebung von Limitierungen in allen Bereichen incl. SQL
- Wesentliche Erweiterung der SQL-Syntax
- Neue Datentypen und Datenbankfunktionen
- Docking und Resizing für eigene Formulare
- Diverse Grafikfunktionen und optische Verbesserungen
- Erweitertes Eigenschaftsfenster mit Builder-Integration
- Hintergrundkompilierung mit Syntaxcoloring bei Fehlern

Schwerpunkt der nächsten Version von Visual FoxPro werden neben dem wichtigen Thema Berichtsgenerator die Aufhebung von Beschränkungen in jedweder Hinsicht und die Erhöhung der Entwicklerproduktivität sein. Das verbessert den Wert des Produktes für Entwickler deutlich, ohne in irgendeiner Weise mit Frameworks zu interferieren. Und natürlich können wir deshalb auch schon allererste Ideen für die Version Visual Extend 9.0 unverbindlich auflisten:

- Integration von CursorAdapter und ggf. XMLAdapter
- Weitere Builder für die Automatisierung von Standardaufgaben
- Integration mit der Toolbox

- Unterstützung für weitere Sprachen
- Unterstützung neuer Features im Berichtsdesigner und Reportgenerator
- Verwaltung von Docking-Positionen von Fenstern für Anwender
- Oberflächen-Klassen für Standardfunktionen wie Adressverwaltung
- Neue Features für Endanwender für die Anpassung der Benutzeroberfläche

Da es aber bis zur Fertigstellung von Visual FoxPro 9.0 noch etwas dauern wird, haben wir uns für den Zeitraum bis dorthin andere Bereiche gesucht. Nämlich die Erstellung von weiteren Beispielanwendungen bzw. Fertiglösungen für die Übernahme in das eigene Angebot z.B. in folgenden Bereichen:

- Adressverwaltung
- Fakturierung
- Finanzbuchhaltung
- Online-Shop-Wartung
- Offline-Reader
- Server Service

Aber sowohl Visual FoxPro 9.0 als auch die korrespondierende Version Visual Extend 9.0 liegen noch einige Zeit vor uns, weshalb Sie sich darüber erstmal keine weiteren Gedanken machen sollten, außer, dass Sie sich sicher sein können, dass die jeweiligen Produkte offensichtlich weiterentwickelt werden und diese Tatsache nur zu Ihrem Nutzen sein kann. Nichts anderes sollen solche Produktvorschauen zeigen - und außerdem verhindern, dass Sie Zeit in die Entwicklung von Features stecken, die Sie in der Folgeversion sowieso für lau erhalten. Die Ankündigung einer Folgeversion ist also nützlich, aber sollte keinesfalls irgendeine Art von Hinderungsgrund sein, sich die jeweils aktuelle Version von Visual FoxPro und Visual Extend erst einmal anzueignen und einzusetzen.

2. VFX 8.0 Schnelleinstieg

2.1. Einführung

Visual Extend gehört seit vielen Jahren zu den leistungsfähigsten Zusatzprodukten von Visual FoxPro. Mit Visual Extend (im folgenden Text mit VFX abgekürzt) ist es möglich in wenigen Minuten den Rahmen für eine Visual FoxPro-Applikation voll funktionsfähig zu erstellen. Wenn vor der Applikationsentwicklung bereits eine Datenbank oder ein Datenmodell zur Verfügung stehen, ist es ein Leichtes mit den Buildern von VFX innerhalb kürzester Zeit Bearbeitungsformulare zu erstellen. Lernen wir die wichtigsten Eigenschaften von VFX kennen in dem wir die Arbeitsschritte zur Erstellung einer Applikation durchgehen.

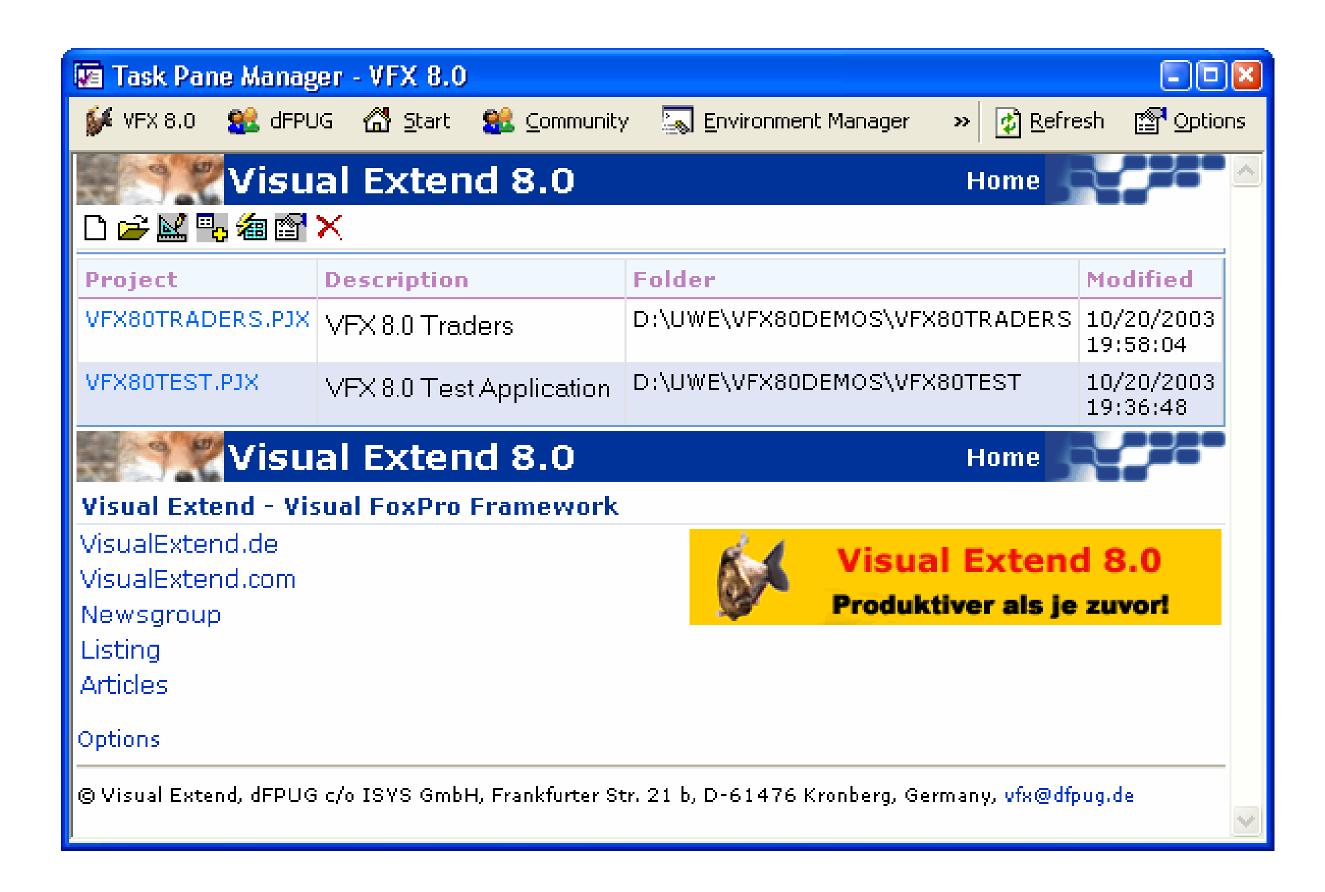
2.1.1. Installation

Nach der Installation von VFX ist es sinnvoll, das VFX-Menü in das Standardmenü von FoxPro zu integrieren. Dazu ist in der Datei Config.fpw eine Zeile einzufügen:

```
Command = DO <VFX-Installationspfad>\builder\vfxmnu.app
```

2.1.2. VFX Task Pane

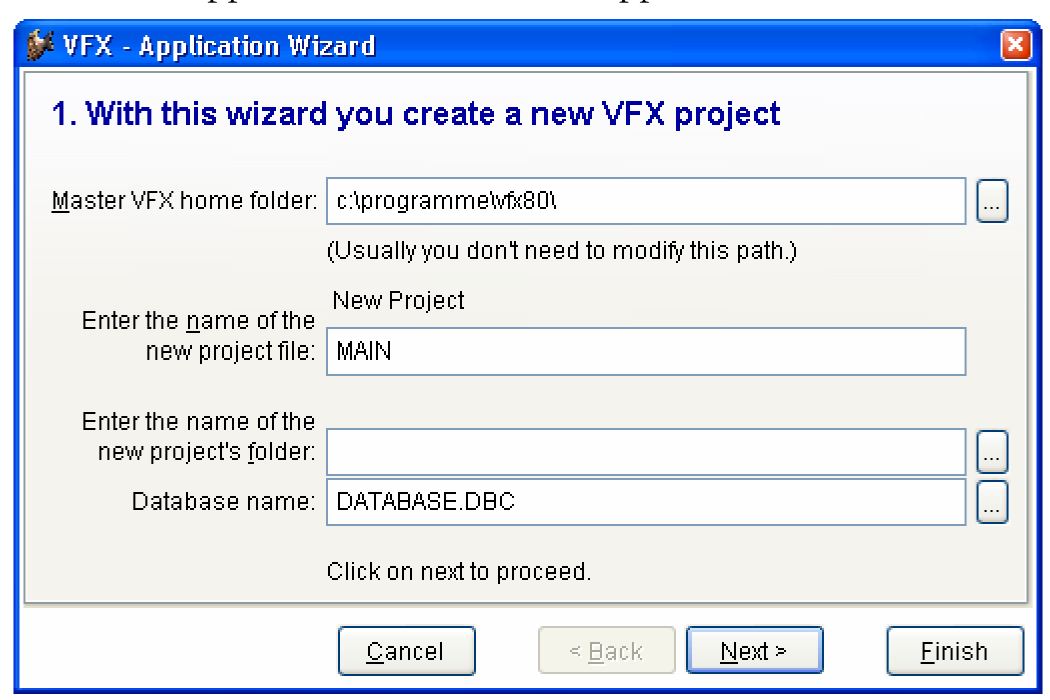
Beim ersten Start von VFP nach der Installation von VFX 8.0 wird automatisch die VFX 8.0 Task Pane integriert. Die VFX 8.0 Task Pane enthält unter anderem alle Funktionen des aus früheren VFX-Versionen bekannten VFX-Application Manager.



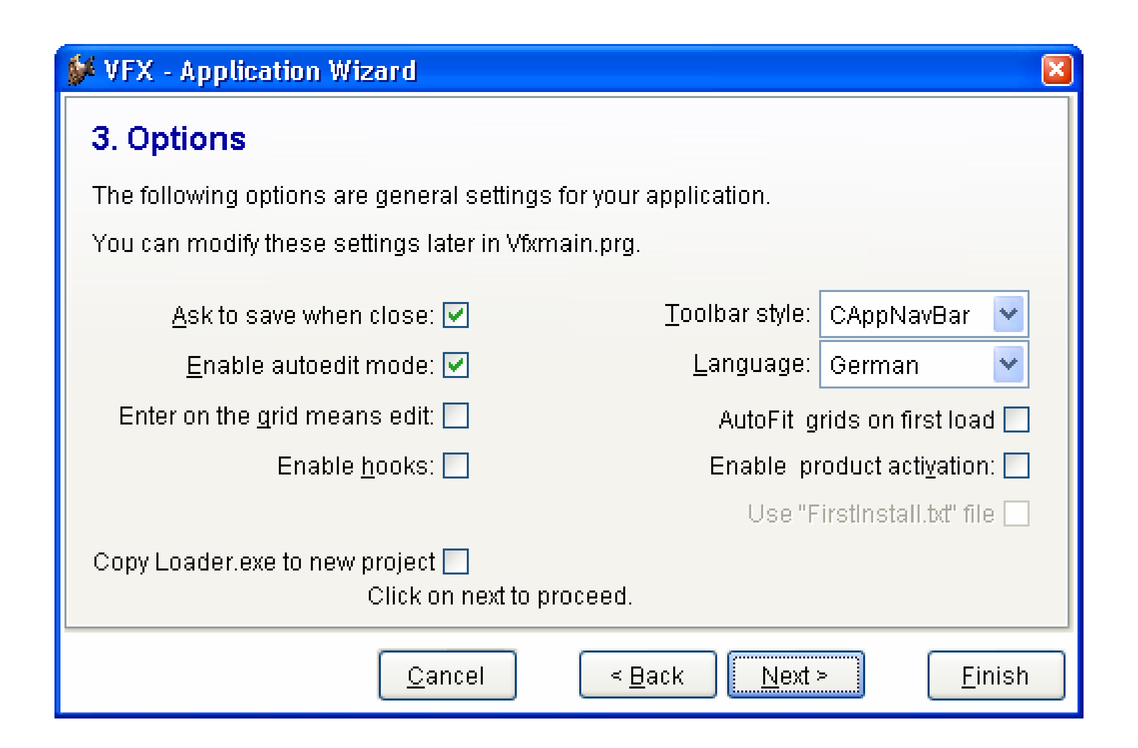
Ein nützliches Tool befindet sich in der VFX 8.0 Task Pane, der Application Manager. In einer Tabelle werden Informationen über alle VFP-Projekte verwaltet. Über den VFX-Application Manager kann ein Projekt geöffnet werden. Dabei wird automatisch der Pfad in das Projektverzeichnis gesetzt. Außerdem kann über den VFX-Application Manager ein „Rebuild all" durchgeführt werden. Dabei wird das Projekt komplett kompiliert. Änderungen in Include-Dateien werden berücksichtigt.

2.1.3. VFX-Application Wizard

Eine neue Applikation wird mit dem Application Wizard erstellt.



Als Sprache für die zu erstellende Applikation wird standardmäßig die Sprache der verwendeten FoxPro-Version vorgeschlagen. Nachdem die „Finish"-Schaltfläche gedrückt wird, werden aus der leeren VFX-Musterapplikation die Dateien in das neu erstellte Projektverzeichnis kopiert und anschließend kompiliert.



2.2. Funktionsumfang der neuen Applikation

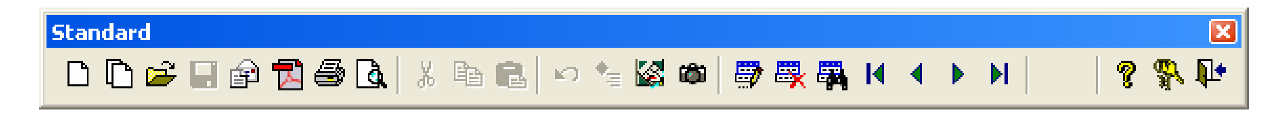
Die mit dem Application Wizard erstellte Applikation kann sofort getestet werden. Dazu kann direkt aus dem Projekt-Manager das Hauptprogamm Vfxmain.prg gestartet werden. Wahlweise kann auch eine App- oder Exe-Datei erstellt und getestet werden. Dies ist während der Entwicklung normalerweise aber nicht erforderlich.

Die Applikation startet mit einem Splashscreen. Als Bild für den Splashscreen wird eine Png-Datei verwendet, die der Entwickler leicht bearbeiten oder austauschen kann. Es ist möglich den Anmeldebildschirm zu umgehen. Nach Anzeige des Splashscreens baut sich der Hauptbildschirm auf und es erscheint der Anmeldebildschirm. Standardmäßig muss sich jeder Benutzer einer VFX-Applikation mit einem Namen und einem Kennwort anmelden. Es ist möglich den Anmeldebildschirm zu umgehen und den Benutzer automatisch mit dem Windows-Anmeldenamen anzumelden. Wahlweise kann die Benutzerverwaltung ganz abgeschaltet werden.

2.2.1. Bedienung

Nach der Anmeldung wird die VFX-Applikation ähnlich den Office-Anwendungen bedient. Benutzer, denen die Bedienung von Word oder Excel geläufig ist, können mit einer VFX-Applikation praktisch sofort produktiv arbeiten.

2.2.2. Standard-Symbolleiste



Alle in der Abbildung nicht beschrifteten Schaltflächen der Symbolleiste sind in ihrer Funktion mit denen aus Office-Produkten identisch.

2.2.3. Formulare

Wenn für ein Formular die lAutoedit-Eigenschaft auf wahr eingestellt ist (das ist der Standardwert), sind ständig alle Steuerelemente auf dem Formular aktiviert. Der Anwender kann mit der Maus oder der Tastatur ein Steuerelement anwählen und sofort mit dem Bearbeiten der Daten beginnen. Das Formular wechselt automatisch in den Bearbeitungsmodus, sobald Daten interaktiv verändert werden.

Auf der Listenseite von VFX-Formularen befindet sich ein Grid. Standardmäßig kann in allen Spalten des Grid inkrementell gesucht werden. Dazu ist einfach der Fokus in die gewünschte Spalte zu setzen. Mit dem ersten Buchstaben- oder Ziffernдruck wird die Sortierfolge auf diese Spalte umgestellt. Dabei wird bei Bedarf automatisch ein temporärer Index erstellt. Die Überschrift in der Spalte wird mit einem auf- oder absteigenden Pfeil, ähnlich dem Windows-Explorer, gekennzeichnet.

Standardmäßig kann die Größe von VFX-Formularen vom Anwender zur Laufzeit geändert werden. Alle Steuerelemente werden dabei proportional in der Größe geändert. Innerhalb von Grids wird die Größe der Steuerelemente standardmäßig nicht verändert. Wenn ein Formular vergrößert wird, werden also mehr Zeilen und Spalten im Grid sichtbar.

Alle Einstellungen an Formularen werden benutzerspezifisch gespeichert. Wenn der Anwender das Formular erneut öffnet, erscheint das Formular an der Position des Bildschirms und in der Größe in der es zuletzt geschlossen

wurde. Auch die Einstellungen der Grids (Spaltenbreiten, Spaltenfolge und Sortierung) werden gespeichert.

VFX-Formulare haben normalerweise eine private Datensitzung und können problemlos mehrfach geöffnet werden. Über eine Eigenschaft des Formulars (lMultiinstance) kann der mehrfache Aufruf verhindert werden.

2.2.4. Benutzerverwaltung

In VFX ist eine Benutzerverwaltung enthalten. Dazu gehören ein Formular zur Bearbeitung der Benutzerdaten, ein Formular zur Bearbeitung der Benutzerrechte und ein Anmeldebildschirm. Über ein numerisches Feld kann eine Benutzerstufe eingestellt werden.

Für alle Felder des aktuellen Benutzer-Datensatzes (aus der Tabelle *Vfxusr.dbf*) der dem angemeldeten Benutzer gehört, werden globale Variablen mit dem Präfix *gu_* angelegt. Es ist an jeder Stelle im Programm möglich, den Wert dieser globalen Variablen abzufragen um zu entscheiden, ob ein Benutzer eine bestimmte Aktion ausführen darf. So kann z. B. die Auswahl eines Menüpunkts, das Öffnen eines Formulars oder das Bearbeiten eines Feldes auf einem Formular verhindert werden.

2.2.5. Fehlerprotokoll

Sollte es einmal zu einem Laufzeitfehler kommen, wird der Fehler in einer Messagebox angezeigt. Außerdem wird der Fehler in einer Tabelle protokolliert. Dabei werden der Name des aktuellen Benutzers, Datum, Uhrzeit, der Status aller geöffneten Tabellen sowie die Ausgabe von List Memory gespeichert. Weitere Eigenschaften der Behandlung von Laufzeitfehlern können über Eigenschaften des Anwendungsobjekts eingestellt werden.

2.2.6. Datenbankwartung

Über den Menüpunkt System – Datenbankwartung wird ein Formular mit einem Mover-Dialog angezeigt.



Hier können Tabellen gepackt oder indiziert werden.

2.2.7. Öffnen-Dialog

Formulare werden standardmäßig über den Öffnen-Dialog gestartet. Der Öffnen-Dialog erscheint im Windows XP-Layout. Die Daten der Formulare stehen in der Tabelle *Vfxfopen.dbf*.

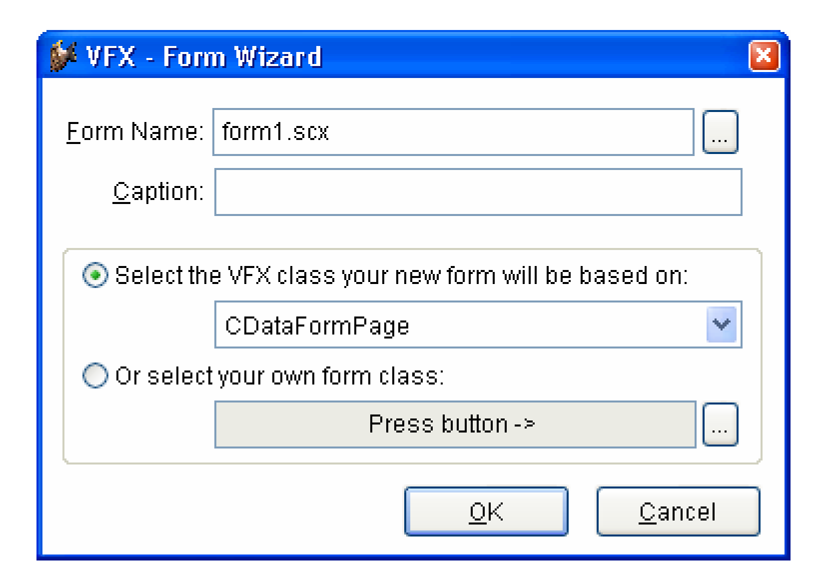


2.2.8. Info-Dialog

Ein Standard-Info-Dialog ist in allen VFX-Applikationen enthalten. Die angezeigten Parameter stammen aus einer Include-Datei, die beim Anlegen des Projektes erzeugt wurde.

2.3. Erstellen eines Formulars mit dem VFX-Form Wizard

Mit Hilfe des VFX-Form Wizard wird ein neues Formular auf der Basis einer VFX-Formularklasse angelegt, in das Projekt eingetragen und zum Bearbeiten geöffnet.



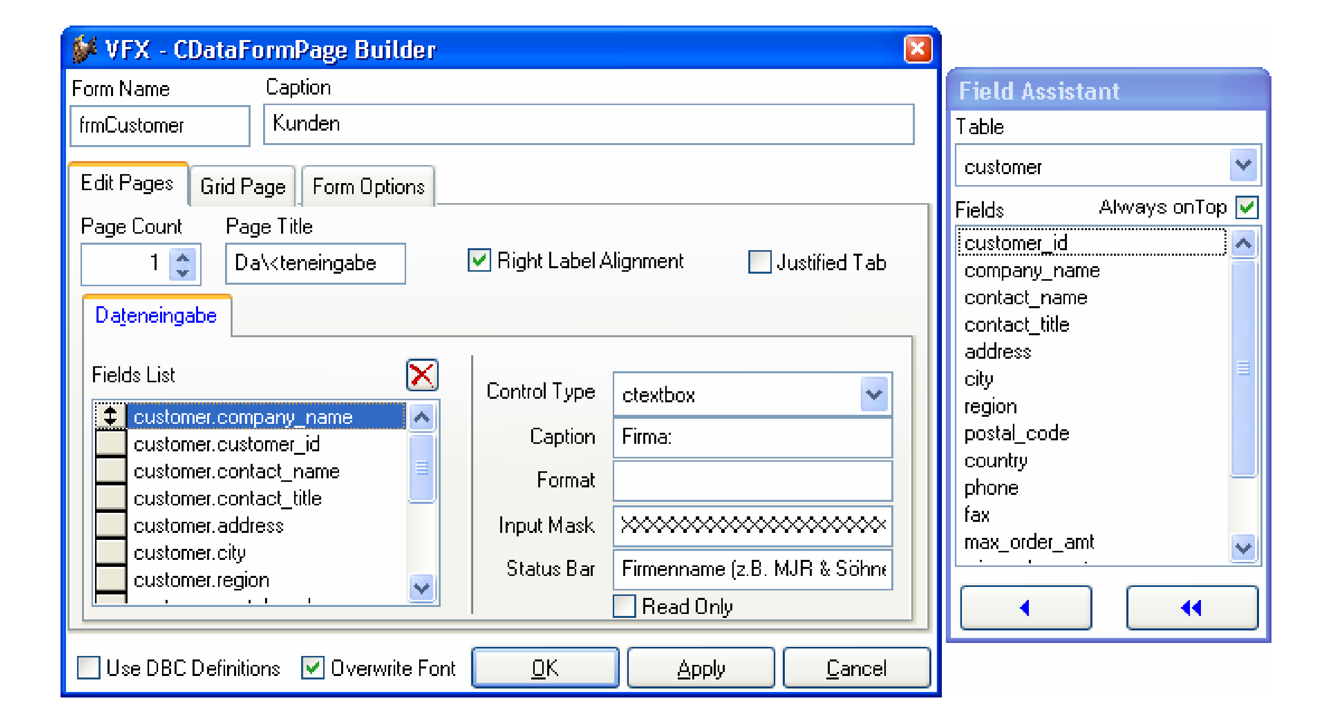
2.4. Datenumgebung

Die von dem Formular zu verwendenden Tabellen oder Ansichten sind in der Datenumgebung einzutragen. Der VFX-Form Builder liest die Datenumgebung aus und stellt die Felder der Tabellen zur Auswahl um Steuerelemente zu erstellen. Zur Laufzeit wird die Datenumgebung ebenfalls ausgelesen um die Tabellen zu ermitteln, für die ein Tableupdate bzw. Tablerevert durchgeführt werden muss.

2.5. Der VFX-Form Builder

Mit diesem Builder werden die für das Formular benötigten Steuerelemente erstellt. Für jedes Steuerelement können dabei die zugrunde liegende VFX-Klasse gewählt sowie einige Eigenschaften eingestellt werden.

Beim ersten Erstellen des Formulars wird automatisch ein Eintrag in der Tabelle Vfxfopen.dbf angelegt, sodass das Formular über den Öffnen-Dialog gestartet werden kann.



Der VFX-Form Builder ist voll reentrant. Das heißt, man kann den Builder beliebig oft aufrufen um Einstellungen am Formular zu verändern. Es ist auch möglich das Formular von Hand mit VFP zu bearbeiten und anschließend wieder mit dem Form Builder zu arbeiten, ohne dass Einstellungen verlorengehen oder überschrieben werden.

2.6. Der VFX-CGrid Builder

Sollen nur Änderungen am Grid vorgenommen werden, braucht nicht der Form Builder verwendet zu werden. Mit dem VFX-Grid Builder können die Einstellungen des Grids verändert werden. Wie alle VFX Builder ist auch der Grid Builder reentrant.

2.7. Test

Das Formular kann direkt aus dem Formular-Designer oder aus dem Projekt-Manager gestartet und getestet werden. In der Init-Methode aller VFX-Formulare wird geprüft, ob das Applikationsobjekt existiert. Falls dieses nicht vorhanden ist, wurde das Formular direkt aus dem Projekt-Manager gestartet und VFX stellt selbständig die Umgebung her, um das Formular voll funktionsfähig laufen zu lassen. Dabei wird auch die Hauptsymbolleiste instanziiert und kann für die Bedienung des Formulars verwendet werden.

Natürlich ist es auch möglich das Projekt über das Hauptprogramm Vfxmain.prg zu starten. Das Formular kann dann über den Öffnen-Dialog gestartet werden.

2.8. Erstellen eines OneToMany Formulars (1:n)

OneToMany-Formulare sehen im oberen Teil des Fensters genauso wie normale Formulare aus. Im unteren Teil befindet sich ein Child-Grid, in dem Daten aus einer Child-Tabelle bearbeitet werden können. Es ist möglich auf einem Seitenrahmen mehrere Child-Grids zu platzieren. Die Grids können dabei verschiedene Spalten der gleichen Tabelle oder Daten verschiedener Child-Tabellen anzeigen.

2.8.1. Der VFX-COneToMany Builder

Zusätzlich zum normalen Form Builder hat der VFX-COneToMany Builder eine Seite zum Erstellen von Child-Grids. Es wird die dem Child-Grid zugrunde liegende Tabelle gewählt und die Spalten werden zusammengestellt.

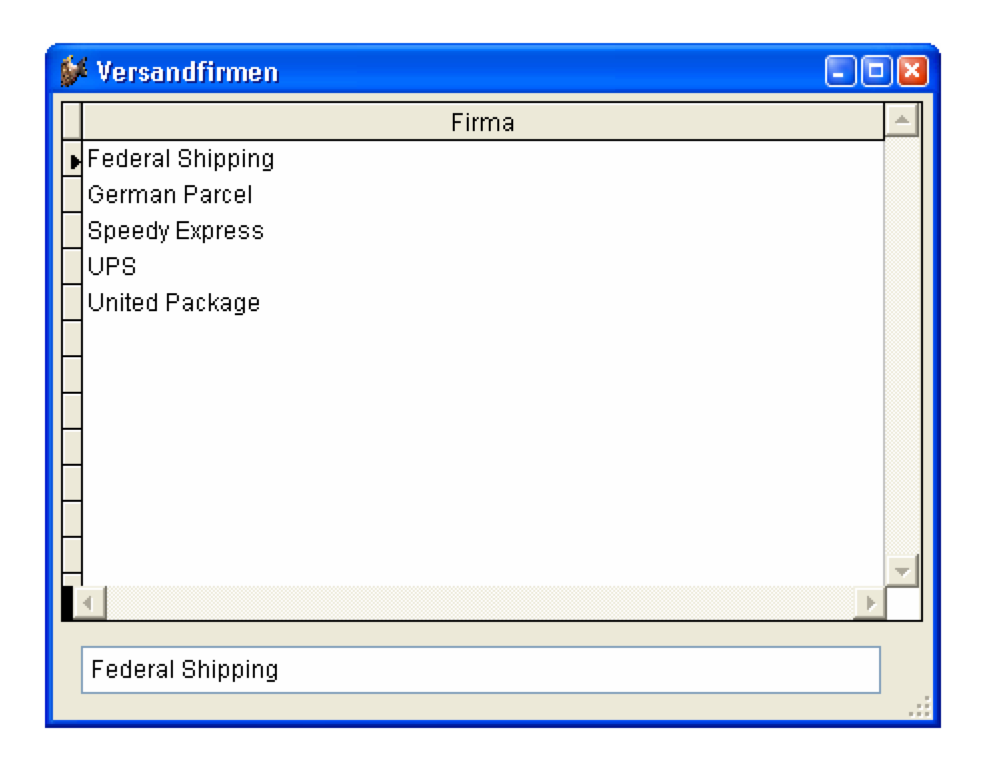
2.8.2. Der VFX-CChildgrid Builder

Im Builder für Child-Grids können die Daten eines Child-Grids bearbeitet werden. Der Unterschied zum normalen Grid Builder besteht darin, dass mit dem Builder für Child-Grids der Code der OnPostInsert-Methode bearbeitet werden kann. Damit ein neuer Child-Datensatz zu einem Parent-Datensatz gespeichert werden kann, muss der Schlüssel des Parent-Datensatzes im Child-Datensatz gespeichert werden. Dies geschieht in der OnPostInsert-Methode. VFX trägt den Code fertig in die OnPostInsert-Methode ein. Der Code wird jedoch in Kommentarzeichen gesetzt. Nach Prüfung durch den Programmierer können die Kommentarzeichen entfernt werden. Eine Nach-

bearbeitung des Codes ist in der Regel nur erforderlich, wenn zusammengesetzte Schlüssel verwendet werden.

2.9. CTableForm

Eine weitere Formularart ist die CTableForm. Bei diesem Formular werden das Listen-Grid und die Steuerelemente nebeneinander oder untereinander dargestellt. Es eignet sich daher insbesondere für Formulare mit nur wenigen Eingabefeldern.



2.10. Weitere Funktionen

Über eine Formulareigenschaft (*lMore*) kann die Schaltfläche „weitere Funktionen" in der Standard-Symbolleiste aktiviert werden. In der Click-Methode dieser Schaltfläche wird die OnMore-Methode des aktiven Formulars aufgerufen. In dieser Methode steht bereits ein Template-Code, der leicht verändert werden kann. Hier werden in einem Array die Parameter für das VFXMore-Formular aufgerufen in dem in einem Dialog zwischen den zur Verfügung stehenden Funktionen ausgewählt werden kann. Z. B. können Child-Formulare gestartet werden.

2.11. Herstellen von Parent/Child-Beziehungen zwischen Formularen

Durch Einstellen von wenigen Eigenschaften in der OnMore-Methode eines Parent-Formulars kann ein Child-Formular gestartet werden. Dem Child-Formular wird der Schlüssel des Parent-Formulars übergeben. Im Child-Formular sind nur die Daten sichtbar, die dem Schlüssel des Parent-Datensatzes entsprechen. Der im Child-Formular sichtbare Bereich kann wahlweise mit einem Filter oder einer Ansicht eingeschränkt werden.

Durch Einstellen einiger Eigenschaften in der OnSetChildData-Methode des Parent-Formulars wird aus dem einfachen Child-Formular ein Linked Child-Formular. Das heißt, wenn im Parent-Formular der Satzzeiger bewegt wird, wird automatisch die Ansicht im Child-Formular entsprechend dem Parent-Schlüssel aktualisiert.

Es ist möglich von einem Parent-Formular mehrere Linked Child-Formulare gleichzeitig zu steuern. Als Formulartyp kommen sowohl für das Parent-Formular als auch für das Child-Formular alle VFX-Formulartypen in Frage. Es ist möglich eine 1:n:m-Beziehung zu realisieren, indem als Linked Child ein OneToMany-Formular verwendet wird.

2.12. Auswahllisten

VFX enthält mehrere Klassen für Auswahlfelder. Ein Auswahlfeld besteht aus einem Textfeld, einer Schaltfläche und einem schreibgeschützten Textfeld. In dem Textfeld kann ein Wert eingetragen werden. Beim Verlassen des Feldes wird überprüft, ob der eingegebene Wert in der Tabelle mit den Auswahlwerten enthalten ist. Falls nein, wird ein Auswahlformular gestartet. Im Auswahlformular kann der Anwender den gewünschten Datensatz auswählen. In einem schreibgeschützten Textfeld können weitere Informationen aus der Auswahltabelle angezeigt werden. Auf Wunsch kann dem Benutzer erlaubt werden neue Datensätze in der Auswahltabelle zu erfassen. Alle Eigenschaften des Auswahlfeldes können mit dem VFX-CPickField Builder gemacht werden.

2.13. Auswahllisten in Childgrids von OneToMany-Formularen

Auch innerhalb von Child-Grids auf 1:n-Formularen können Auswahllisten verwendet werden. Die Einstellungen werden auch hierfür über einen speziellen Builder gemacht.

2.14. Hinzufügen einer Symbolleiste zu einem Formular

Sehr anwenderfreundlich ist die Möglichkeit einem Formular eine Symbolleiste hinzuzufügen. Die Symbolleiste wird normal mit VFP erstellt. In der Click-Methode der Symbolleisten-Schaltflächen wird Sinnvollerweise eine Methode des aktiven Formulars aufgerufen. Z. B.:

```
_screen.activeform.meinemethode()
```

Der Name der Symbolleiste wird in einer Eigenschaft des Formulars *ctoolbarclass* eingetragen. VFX zeigt die Symbolleiste automatisch an, wenn das Formular aktiv ist und versteckt sie wieder, wenn ein anderes Formular aktiv wird. Selbstverständlich werden der Status und die Position der Symbolleiste benutzerspezifisch gespeichert.

2.15. Client/Server-Anwendungen

2.15.1. Verwenden von Ansichten

Als Datenquelle für ein Formular kann wahlweise eine Tabelle oder eine Ansicht verwendet werden. Wird eine Ansicht verwendet, muss beim Formular die Eigenschaft *lWorkOnView* auf .T. gestellt werden.

Ansichten können für jeden VFX-Formulartyp als Datenquelle verwendet werden. Es ist möglich OneToMany-Formulare oder Parent/Child-Konstruktionen auf Ansichten basieren zu lassen. Auch ist die Verwendung von Ansichten bei Auswahllisten möglich. Eine VFX-Anwendung kann somit als Frontend z. B. für einen SQL-Server oder andere Remote-Datenquellen verwendet werden.

2.15.2. Eingabe der Ansichtsparameter

Zur Eingabe der Ansichtsparameter gibt es eine spezielle VFX-Formularklasse. Es wird ein Formular auf Basis der Klasse *cAskViewArg* erstellt. Vom Bearbeitungsformular können die Eingabefelder, die Ansichtsparameter enthalten, über die Zwischenablage auf das Formular zur Eingabe der Ansichtsparameter kopiert werden. In einer Eigenschaft (*cviewparameter*) des jeweiligen Eingabefeldes wird der Name des Ansichtsparameters eingetragen. Das Formular zur Eingabe der Parameter kann in der Init-Methode des Bearbeitungsformulars oder z. B. über eine Schaltfläche aufgerufen werden.

2.16. Verändern von Eigenschaften des Applikationsobjektes

Im Hauptprogramm Vfxmain.prg wird programmatisch eine Ableitung der Klasse des Applikationsobjektes erstellt. Hier ist es also möglich den Code der VFX-Methoden zu ändern und Eigenschaften einzustellen ohne an den Klassenbibliotheken Veränderungen vornehmen zu müssen.

2.17. Mover-Dialog

Der Mover-Dialog ist ein praktisches Werkzeug zur Auswahl von relativ wenigen Daten. Die Mover-Klasse wird programmatisch instanziiert. Parameter sind ein Array mit der Auswahlliste und ein Array mit den ausgewählten Elementen, das nach Beenden des Dialogs auch die Ergebnismenge enthält.

2.18. OLE-Klassen

Es ist möglich Word, Excel, Outlook und Powerpoint per OLE aus VFX-Applikationen anzusteuern. Die wichtigsten Funktionen stehen in Klassen zur Verfügung.

2.19. Debug-Modus

Durch setzen einer Konstanten kann die Anwendung im Debug-Modus gestartet werden. Im Debug-Modus ist ein zusätzliches Menü sichtbar, mit dessen Hilfe jederzeit der Debugger gestartet werden kann. Außerdem kann durch einen Rechtsklick mit der Maus auf einem Formular der Debugger gestartet werden. Dabei wird auch das Set-Fenster geöffnet.

2.20. Systemeinstellungen im Optionen-Dialog

Im Optionen-Dialog können die Felder der Tabelle *Vfxsys.dbf* bearbeitet werden. Der Programmierer kann dieser Tabelle Felder mit globalen Einstellungen hinzufügen. Zur Laufzeit stehen die Werte aller Felder als globale Variablen mit dem Präfix *gs_* zur Verfügung.

2.21. Mehrsprachige Applikationen, VFX-LangSetup Builder

Bei der Erstellung eines neuen VFX-Projekts kann zwischen den Sprachen deutsch, englisch, französisch italienisch, spanisch, griechisch, bulgarisch und tschechisch gewählt werden. Entsprechend zur gewählten Sprache werden die Include-Dateien in das neue Projekt kopiert.

Will man zu einem späteren Zeitpunkt seine Applikation in eine andere Sprache übersetzen, startet man für jedes Formular den VFX-LangSetup Builder. Dieser Builder erstellt für jede Caption eines Formulars eine Zuweisung. Der Caption wird zur Laufzeit der Wert einer Konstanten zugewiesen.

Die Konstanten können mit dem VFX-Message Editor erstellt und bearbeitet werden. Zur Erstellung der Applikation kopiert man dann einfach die Include-Dateien der gewünschten Sprache in das Projekt und lässt die Anwendung erstellen.

2.22. Aktualisierung der Kundendatenbank

VFX enthält Routinen um eine Aktualisierung der Datenbank beim Kunden automatisch durchzuführen. Dazu wird unterhalb des Datenverzeichnisses ein Verzeichnis mit dem Namen Update angelegt. In dieses Verzeichnis wird die Datenbank mit allen Tabellen, jedoch ohne Daten, kopiert. Beim Programmstart wird die Datenbank im Datenverzeichnis aktualisiert. Es können der Datenbank auf diese Weise neue Tabellen, neue Felder in Tabellen, neue Indexschlüssel und neue Ansichten hinzugefügt werden. Ebenso werden nicht mehr benötigte Tabellen, Felder usw. gelöscht. Anschließend werden alle Dateien im Update-Verzeichnis gelöscht. Mit dieser Methode können auch freie Tabellen aktualisiert werden.

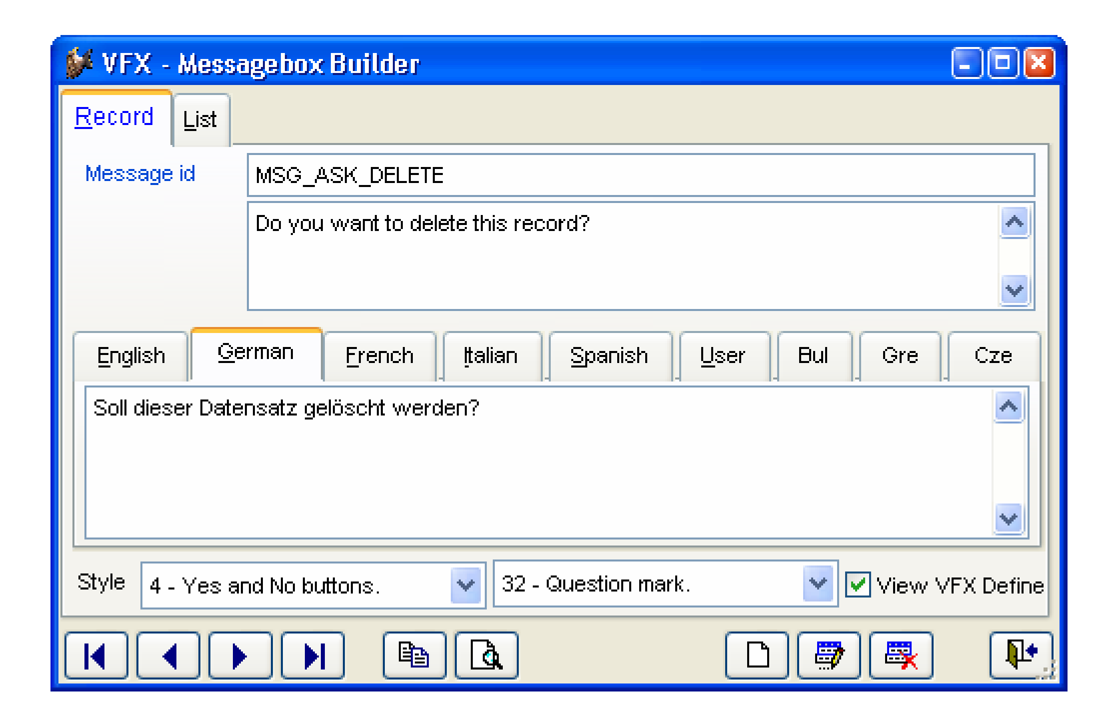
In VFX 8.0 werden auch die Erstellung und Aktualisierung von SQL Server-Datenbanken unterstützt.

2.23. VFX-Class Switcher

Mit dem VFX-Class Switcher ist es möglich nachträglich die einem Steuerelement zugrunde liegende Klasse zu ändern. So kann man z. B. aus einer Textbox einen Spinner oder ein Auswahlfeld machen.

2.24. VFX-Messagebox Builder

Ein nützliches Werkzeug zur Erstellung von Messageboxen in verschiedenen Sprachen ist der VFX-Message Box Builder. Die Texte der Messagebox werden in der Tabelle *Vfxmsg.dbf* gespeichert. Der Befehl zur Anzeige der Messagebox wird in die Zwischenablage kopiert und kann von dort in den eigenen Programmquelltext übernommen werden. Dabei wird nicht der Text selbst, sondern eine Konstante als Parameter übergeben. Die Include-Dateien mit den Werten der Konstanten in der gewünschten Sprache werden mit dem VFX-Message Editor erstellt.



2.25. VFX-Message Editor

Die Werte aller von VFX verwendeten Konstanten stehen in der freien Tabelle *Vfxmsg.dbf*. Für jede Sprache ist ein Memofeld mit dem Text vorhanden. Mit dem VFX-Message Editor können diese Texte bearbeitet werden.

2.26. Hooks

VFX bietet bei allen wichtigen Methoden Eingriffsmöglichkeiten über Hooks. Als Beispiel schauen wir die OnInsert-Methode eines Formulars an. Die OnInsert-Methode wird aufgerufen, wenn ein neuer Datensatz angefügt werden soll. Dabei wird zunächst die Methode OnPreInsert aufgerufen. Nur wenn diese Methode .T. als Rückgabewert liefert, wird ein Datensatz angefügt. Nach dem Anfügen des Datensatzes wird die OnPostInsert-Methode aufgerufen. Hier können z. B. mit dem Replace-Befehl Daten in den neuen Datensatz eingetragen werden. Wenn die OnPostInsert-Methode .F. zurückliefert, wird ein Tablerevert() durchgeführt und der neue Datensatz damit sofort wieder gelöscht.

Zusätzlich zu diesen Möglichkeiten ist in den meisten VFX-Methoden ein Eventhook eingebaut. Wenn die Eventhooks aktiviert sind, wird in jedem Eventhook die Funktion Eventhook-Handler aufgerufen. Als Parameter werden dieser Funktion der Name der aufrufenden Methode, eine Referenz auf das aktuelle Objekt und eine Referenz auf das aktuelle Formular übergeben. Über eine Case-Konstruktion kann dann individueller Code ausgeführt werden. Hierdurch kann an praktisch jeder Stelle in den Funktionsablauf von VFX eingegriffen werden.

3. Einführung

3.1. Überblick

Visual Extend stellt eine umfassende Entwicklungsumgebung für Softwareentwickler dar, die mit Microsoft Visual FoxPro 8.0 oder einer neueren Version arbeiten. Visual Extend beinhaltet Builder, die den Softwareentwickler bei seiner täglichen Arbeit unterstützen und so die Entwicklerproduktivität drastisch steigern. Dies, ohne jegliche Einbußen bezüglich Flexibilität oder Leistungsfähigkeit in Kauf nehmen zu müssen. Visual Extend macht aus Visual FoxPro ein echtes Rapid Application Development Tool, dies sowohl für Desktop- als auch für Client/Server-Datenbank-Applikationsentwicklungen.

Visual FoxPro ist ein exzellentes Entwicklungswerkzeug. Dank der Objektorientierung und der OLE-Technologie wird der Traum eines jeden Softwareentwicklers nach einfachster Wiederverwendung von eigenen oder fremden Softwaremodulen Wirklichkeit. Das Erstellen einer eigenen Entwicklungsumgebung stellt jedoch ein größeres Unterfangen dar, welches sich heutzutage immer weniger Softwareentwickler wirklich leisten können. Es ist nicht nur schwierig, eine stabile Klassenbibliothek für alle Anwendungen zu entwickeln, es wäre auch sehr zeitaufwendig, die Klassen manuell einzusetzen und alle Eigenschaften und Methoden über das Eigenschaftsfenster während der Entwicklung einer neuen Anwendung zu bearbeiten.

Visual Extend für Visual FoxPro füllt exakt diese Lücke und stellt eine vollständige Anwendungsentwicklungsumgebung für Visual FoxPro Softwareentwickler dar. Dank des durchdachten, modularen Designs von Visual Extend kann der Softwareentwickler jederzeit selbst entscheiden, ob er die gesamte Entwicklungsphilosophie von Visual Extend verwenden oder nur ausgewählte Teile daraus für die Erstellung seiner eigenen Anwendungen übernehmen will. Die Objektorientierung von Visual Extend erlaubt dem Entwickler Unterklassen aller Visual Extend Klassen zu erstellen, um so die Entwicklungsumgebung noch besser seinen spezifischen Bedürfnissen anzupassen.

Visual Extend ist weit mehr als nur eine Klassenbibliothek. Vielmehr beinhaltet Visual Extend neben einer leistungsfähigen Klassenbibliothek ebenso leistungsfähige Builder, um einen maximalen Produktivitätsgewinn zu erzielen. Visual Extend besteht aus den folgenden Hauptkomponenten:

- Modulare den Microsoft Standards entsprechende Klassenbibliothek zur umfassenden Unterstützung bei der Anwendungsentwicklung.
- Visual Extend Assistenten und voll wieder verwendbare Builder für Anwendung, Formular, Grid, Child-Grid, Auswahlliste, Auswahltextfeld und 1:n-Formulare.
- Weitere Visual Extend Entwickler-Produktivitätswerkzeuge wie das Entwicklermenü, die VFX Task Pane, der VFX – Base Class Switcher und der Visual Object Name Picker.

3.2. Eigenschaften von mit Visual Extend erstellten Anwendungen

Anwendungen, die mit Visual FoxPro und der Software-Entwicklungsumgebung Visual Extend entwickelt wurden, haben die folgenden Eigenschaften:

- Bereit zur Office-Compatible-Zertifizierung.
- Standard-Symbolleiste und optionale individuelle Symbolleiste für jedes Formular.
- Unterstützung von XP-Themes in allen Steuerelementen.
- Hot Tracking von Schaltflächen in Symbolleisten.
- Icons in Menüs.
- Navigieren, Suchen, Neu, Kopieren, Bearbeiten, Löschen als Optionen im Formular oder in der Symbolleiste.
- Multiinstanzfähige Formulare.
- Zuletzt aufgerufene Formulare im Menü Datei sowie aktuell geöffnete Formulare im Menü Fenster.
- Inkrementelle Suche inklusive automatischer Sortierung in allen VFX-Grids.
- Wechsel der Sortierung durch Doppelklick auf die Spaltenüberschrift in jedem VFX-Grid.
- Anzeige der aktuellen Sortierung in der Spaltenüberschrift, wahlweise auch farbliche Anzeige.
- Automatisches Speichern und Wiederherstellen der Größe und der Position von allen Formularen.
- Automatisches Speichern und Wiederherstellen aller Layoutänderungen und der aktuellen Sortierfolge im Grid.
- Auswahllisten-Steuerelement mit automatischer Validierung.

- Auswahllisten-Formular mit inkrementeller Suche, automatischer Sortierung, Wechsel der Sortierung durch Doppelklick auf eine Spaltenüberschrift und Start des Bearbeitungsformulars mit der Möglichkeit neue Datensätze einzugeben.
- Automatisches Speichern und Wiederherstellen der Größe und Position von allen Auswahllisten-Formularen inklusive aller Layoutänderungen im Auswahllisten-Grid.
- Leistungsfähige Auswahllisten in Child-Grids.
- Benutzerverwaltung mit Kennwort-Verschlüsselung.
- Automatische Übernahme des Netzwerk-Anmeldenamens und Möglichkeit der automatischen Benutzeranmeldung.
- Verwaltung der Benutzerrechte mit Ansichts-, Bearbeitungs-, Neuanlage- und Löschrecht auf Formularebene.
- Datenbankwartung für das Komprimieren und neu Indizieren von lokalen Tabellen.
- Automatisches protokollieren aller Laufzeitfehler.
- System Lock-Tabelle für optionale Semaphore Locking Schemata.
- Infodialog.
- Benutzerfreundliche Mover-Dialoge für die einfache Auswahl mehrerer Elemente.
- Automatische Übernahme der Windows-Systemfarben.
- Favoriten-Menü.
- Öffnen-Formular im XP-Stil.
- Optionale „Active-Desktop“ Einzelklick-Benutzeroberfläche.
- Automatisches Erstellen von gedruckten Berichten basierend auf der Datenanzeige in einem Grid.
- Berichtsauswahl und –bearbeitungsdialog.
- Unterstützung mehrerer Datenbanken mit der Möglichkeit die Datenbank zur Laufzeit zu wechseln.
- Automatische Aktualisierung der Strukturen der Kundendatenbank für VFP- und SQL Server-Datenbanken.
- Optionales Bearbeitungsprotokoll zur Verfolgung der Datenbearbeitung.
- Die Microsoft Agenten können zur Gestaltung der Benutzeroberfläche verwendet werden.
- Automatischer Ausdruck des Bildschirminhalts.
- Es können mehrsprachige Anwendungen erstellt werden.

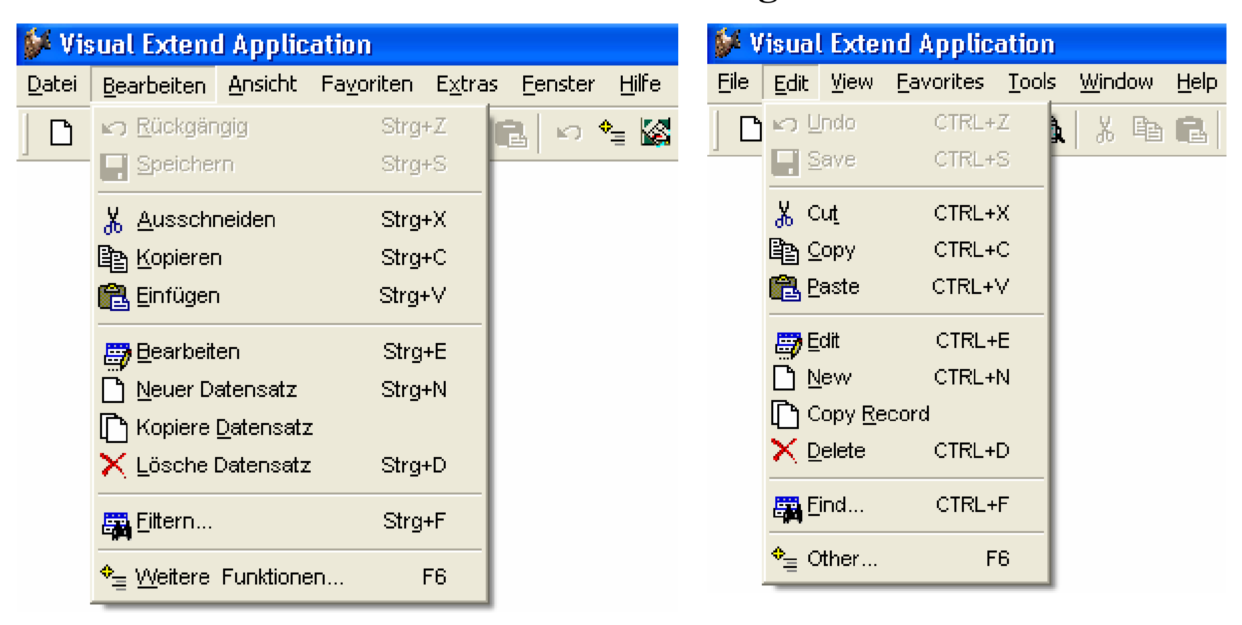
3.3. Leistungsmerkmale

Softwareentwickler werden die folgenden Visual Extend-Merkmale besonders zu schätzen wissen:

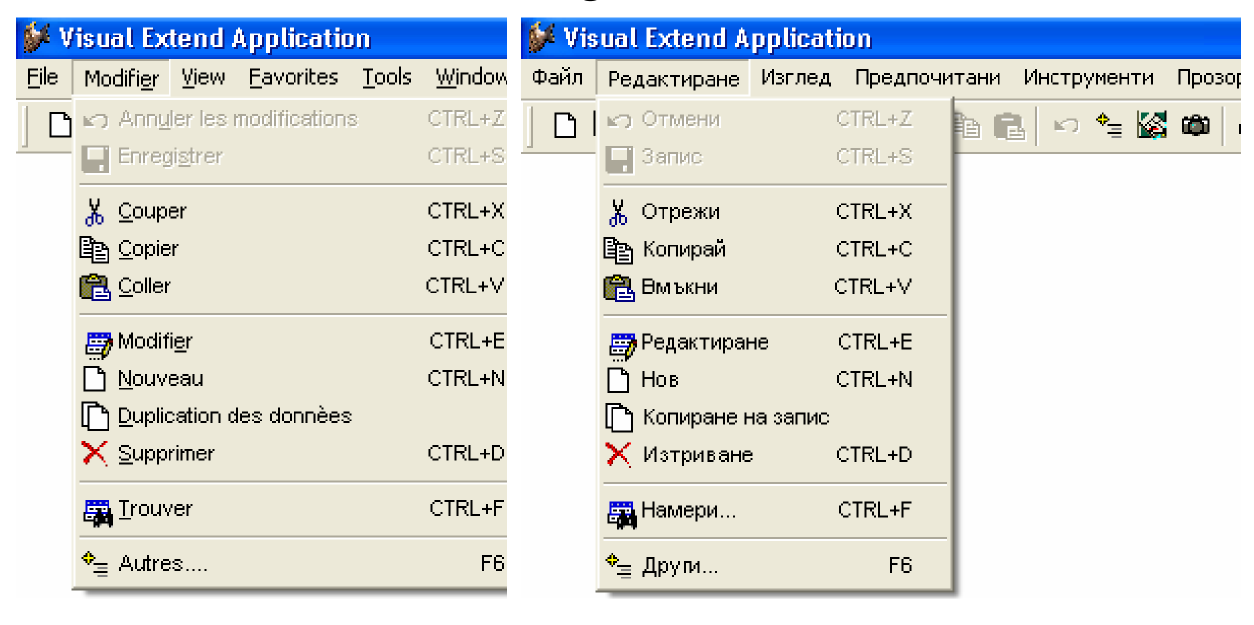
- Anwendungs-Assistent für das automatische Erstellen von neuen Anwendungen in der Sprache Ihrer Wahl. Nach nur wenigen Sekunden ist Ihre lauffähige Visual FoxPro Anwendung vorbereitet!
- Volle Widerverwendbarkeit von allen VFX-Buildern (Formular-Builder, 1:n-Formular-Builder, Table Form-Builder, Grid-Builder, Child-Grid-Builder, Auswahltextbox-Builder), die es vereinfachen, Änderungen an mit den VFX-Buildern erstellten Formularen durchzuführen!
- Benutzen Sie die Visual FoxPro-Entwicklungsumgebung wann immer Sie wollen, ohne die Widerverwendbarkeit der VFX-Builder zu verlieren, solange Sie alle Steuerelemente mit Hilfe der VFX-Builder hinzufügen bzw. entfernen!
- Builder für Standardformulare inklusive Parent/Child-Technik (aufrufen und aufgerufen von).
- Builder für leistungsfähige Grids.
- Builder für jeden Bedarf an Auswahllisten.
- Builder für klassische sowie fortgeschrittene 1:n-Formulare mit mehrseitiger Bearbeitung der Haupttabelle sowie mehrseitiger Bearbeitung für mehrere Child-Tabellen in einem Formular.
- Alle Builder lesen die vorhandenen Feldbeschreibungen und andere Eigenschaften aus der Datenumgebung.
- Die Formular-Builder passen die Längen der Textfeld-Steuerelemente den Größen der zugrunde liegenden Felder an.
- Die VFX-Formular-Builder sind auf eigenen, von den VFX-Klassen abgeleiteten Klassen einsetzbar.
- Testen von Formularen direkt aus dem Formular-Designer.
- Navigieren mit der Symbolleiste oder mit Navigations-Schaltflächen auf dem Formular oder mit Schaltflächenleisten innerhalb eines Formulars.
- Messagebox-Assistent.
- Task Pane Anwendungs-Manager.
- Einfaches Bilden einer Unterklasse der Anwendungsklasse und Anpassen der Umgebungsklasse.
- Einfaches Erstellen der anwendungsspezifischen Standard-Symbolleisten.
- Technik verbundener Child-Formulare.

- Die Entwicklungsumgebung stellt bereits alle Elemente der Benutzeroberfläche in den Sprachen deutsch, englisch, französisch, spanisch, italienisch, bulgarisch, tschechisch und griechisch zur Verfügung. Starten Sie eine neue Anwendung in der Sprache Ihrer Wahl, ohne ein Wort der Visual Extend Software-Entwicklungsumgebung übersetzen zu müssen.

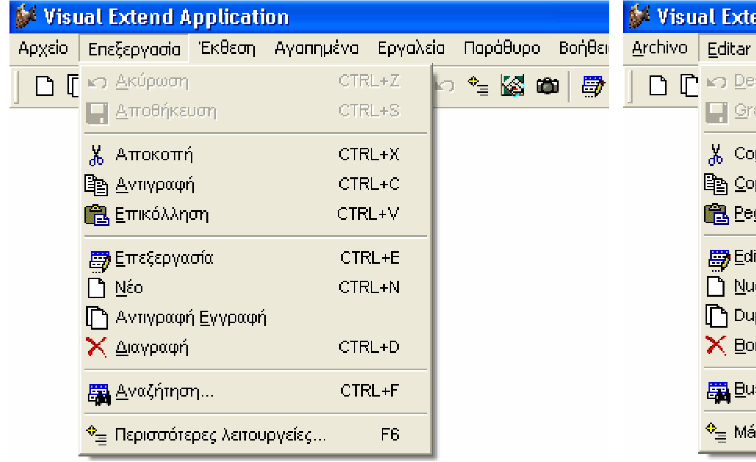
Deutsch **Englisch**



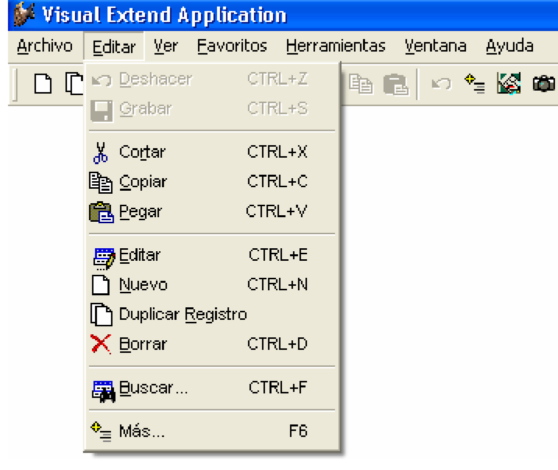
Französisch **Bulgarisch**



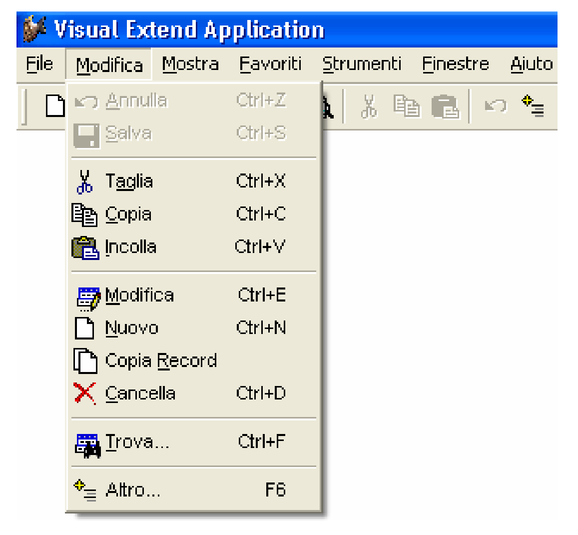
Griechisch



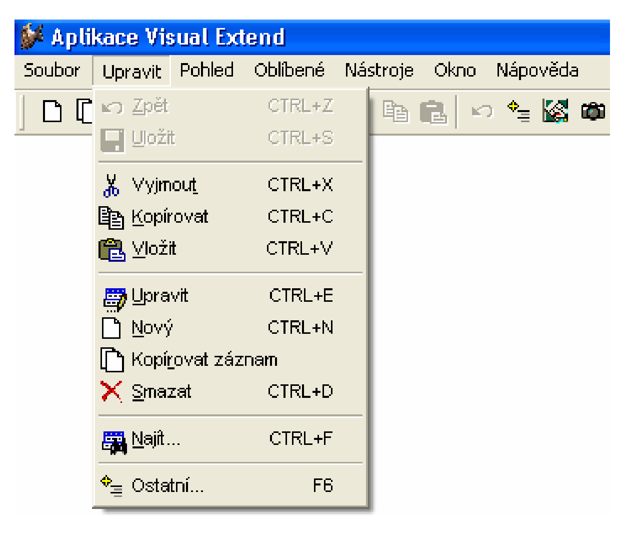
Spanisch



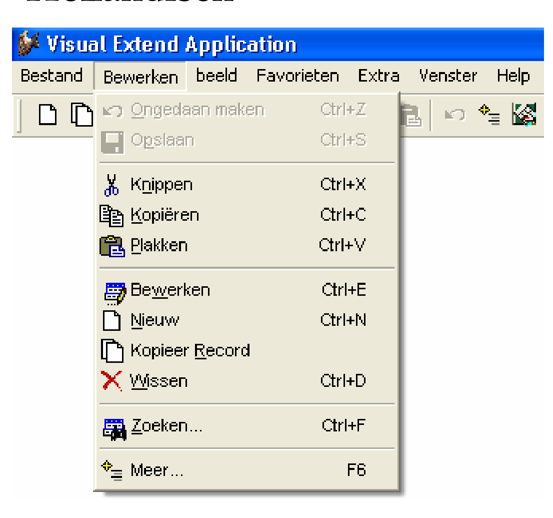
Italienisch



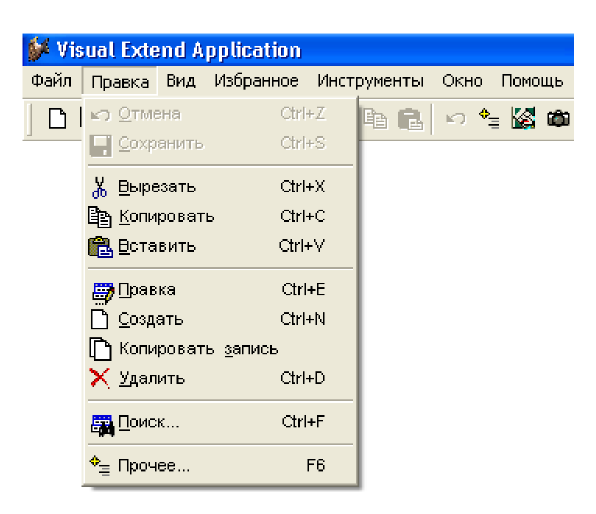
Tschechisch



Holländisch



Russisch



VFX hilft Ihnen, Ihre Visual FoxPro-Anwendungen in einer höheren Qualität und wesentlich schneller als bisher zu entwickeln. Ihre Entwickler-Produktivität steigert sich dramatisch. Und das alles, ohne irgendwelche Einbußen bezüglich der Flexibilität von Visual FoxPro in Kauf nehmen zu müssen. Produktiver als je zuvor mit Visual Extend für Visual FoxPro!

4. Leistungsumfang

4.1. VFX-Klassenbibliothek

Sie finden die Klassenbibliotheken im Ordner *VFX80**LIB*. Um eine detaillierte Beschreibung aller Dateien der Klassenbibliotheken mit allen Klassen, Eigenschaften und Methoden zu bekommen, lesen Sie bitte in der VFX Technische Referenz nach. Die Technische Referenz ist eine Windows-Hilfedatei und kann online gelesen werden.

4.2. VFX-Assistenten und Builder

Alle VFX-Assistenten und Builder befinden sich im Ordner \\VFX80\\BUILDER:

Assistent	Datei	Beschreibung
VFX-Assistenten und Builder	VFXBLDR.APP	Die folgenden VFX-Assistenten und Builder helfen Ihnen bei der Erstellung von professionellen Visual FoxPro-Anwendungen in Rekordzeit: √ Anwendungs-Assistent für die Erstellung einer neuen Anwendung √ Formular-Assistent für die Erstellung eines neuen Formulars √ Formular-Builder (inklusive mehrseitigen Formularen, **wieder verwendbar**) √ Grid-Builder (**wieder verwendbar**) √ Auswahllisten-Builder (**wieder verwendbar**) √ 1:n-Builder (inklusive mehrseitigen Seitenrahmen für die Haupttabelle und mehreren Seiten für die Child-Tabellen, **wieder verwendbar**) √ Child-Grid-Builder (**wieder verwendbar**) √ Auswahllisten-Builder für Auswahllisten innerhalb von Child-Grids (**wieder verwendbar**) *Wenn Sie die Installationsanweisungen befolgen, können Sie mittels rechter Maustaste den VFX-Builder aufrufen, nachdem Sie das entsprechende Objekt ausgewählt haben.*
VFX LangSetup Builder	LANGBLDR.APP	Automatisieren Sie die Erstellung des Codes für die VFX-*LangSetup()*-Methode. Dies ist eine sehr große Hilfe, wenn Sie mehrsprachige Anwendungen erstellen. ***Aufrufen können*** *Sie den LangSetup-Assistenten aus dem VFX-Menü (DO VFXMNU) oder indem Sie LANGBLDR.APP starten*
VFX Messagebox Builder	MSGBLDR.APP	Automatisieren Sie das Generieren von Messagebox-Dialogen und den zugehörigen Konstanten in den Include-Dateien. ***Aufrufen können*** *Sie den Messagebox-Assistenten aus dem VFX-Menü (DO VFXMNU) oder indem Sie MSGBLDR.APP starten.*
VFX Message Editor	MSGEDIT.APP	Automatisieren Sie die Lokalisierung von Meldungen und anderen Texten sowie das Generieren der entsprechenden Include-Dateien. ***Aufrufen können*** *Sie den Message-Editor aus dem VFX-Menü (DO VFXMNU) oder indem Sie MSGEDIT.APP starten.*

Alle VFX 8.0 Formular-, Grid- und Auswahllisten-Builder sind voll wieder verwendbar! Das bedeutet, dass Sie diese Builder im Entwicklungszyklus beliebig oft aufrufen können, ohne zuvor eingegebene Einstellungen zu verlieren. Ebenso werden Änderungen Ihres Formulars, die Sie nach der Generierung mit Visual FoxPro gemacht haben, von den Buildern beim nächsten Aufruf eingelesen.

Durch die offene Architektur der VFX-Assistenten steht fortgeschrittenen Benutzern der von den Assistenten verwendete Code in der Tabelle *\VFX80\LIB\BUILDER\VFXCODE.DBF* zur Verfügung. Dadurch können Sie die Assistenten einfach Ihren eigenen Code verwenden lassen. **Warnung:** Änderungen in dieser Tabelle erfordern fortgeschrittenes Wissen über VFX.

ANMERKUNG: Benutzen Sie die VFX-Builder so lange wie möglich, um Steuerelemente hinzuzufügen oder zu entfernen (definiert durch die ausgewählten Felder). Dadurch profitieren Sie am meisten von der hohen Produktivität, den die Builder bieten!

4.3. VFX-Produktivitätswerkzeuge

Um Ihre Arbeit mit VFX noch produktiver werden zu lassen, stehen Ihnen weitere nützliche Produktivitätswerkzeuge zur Verfügung:

Werkzeug	Datei	Beschreibung
VFX Task Pane	VFXTASKPANE.XML	Die VFX Task Pane erlaubt Ihnen ein problemloses Wechseln zwischen verschiedenen Projekten. Die Tabelle, die die aktuellen Referenzen zu Ihren Projekten speichert, ist VFXAPP.DBF/CDX/FPT. Diese Tabelle befindet sich im BUILDER-Ordner.
VFX Menü	VFXMNU.APP	Richtet einen speziellen Menüpunkt in Ihrem Visual FoxPro Menü ein. Von diesem Menü aus können Sie den VFX-Anwendungs-Assistenten und weitere VFX-Assistenten aufrufen. **Tipp:** Wenn Sie die Installationsanleitung befolgt haben, wird dieses Menü automatisch geladen wenn Sie VFP starten.
VFX Class Switcher	<im VFXBLDR, aus dem VFX-Menü aufzurufen>	Ändert die Klasse aller Formulare. Ermöglicht ein einfaches Wechseln von Formularen ohne Navigationsschaltflächen (z. B.: *CDataFormPage*) zu solchen mit Navigationsschaltflächen (z. B.: *CDataFormPageBar*). Sie können mit dem Class Switcher auch die Klasse eines selektierten Steuerelementes ändern.
VFX Object Name Picker	<im VFXBLDR, aus dem VFX-Menü aufzurufen>	Kopiert die vollständige Referenz des aktuell ausgewählten Steuerelements in die Zwischenablage. Das ist manchmal sehr nützlich, da visueller als die VFP-Objektliste, die Sie mit der rechten Maustaste in einem Codefenster öffnen können.

4.4. Neue Entwicklerwerkzeuge

Zusätzlich zu den schon in früheren VFX-Versionen vorhandenen Buildern stehen in VFX 8.0 neue Power Builder für folgende Klassen zur Verfügung:

- cTreeViewForm
- cTreeViewOneToMany
- cPickAlternate

Zur Unterstützung der Produktaktivierung werden die zwei Assistenten benötigt: (Siehe auch: Applikationsschutz durch Produktaktivierung)

- Define Activation Rules – Einstellen der Systemeigenschaften, die zur Produktaktivierung verwendet werden sollen sowie der möglichen Benutzerrechte.
- Create Activation Key – Erstellen eines Aktivierungsschlüssels anhand des Installationsschlüssels des Kunden.

Assistent zum Anlegen von SQL Metadaten:

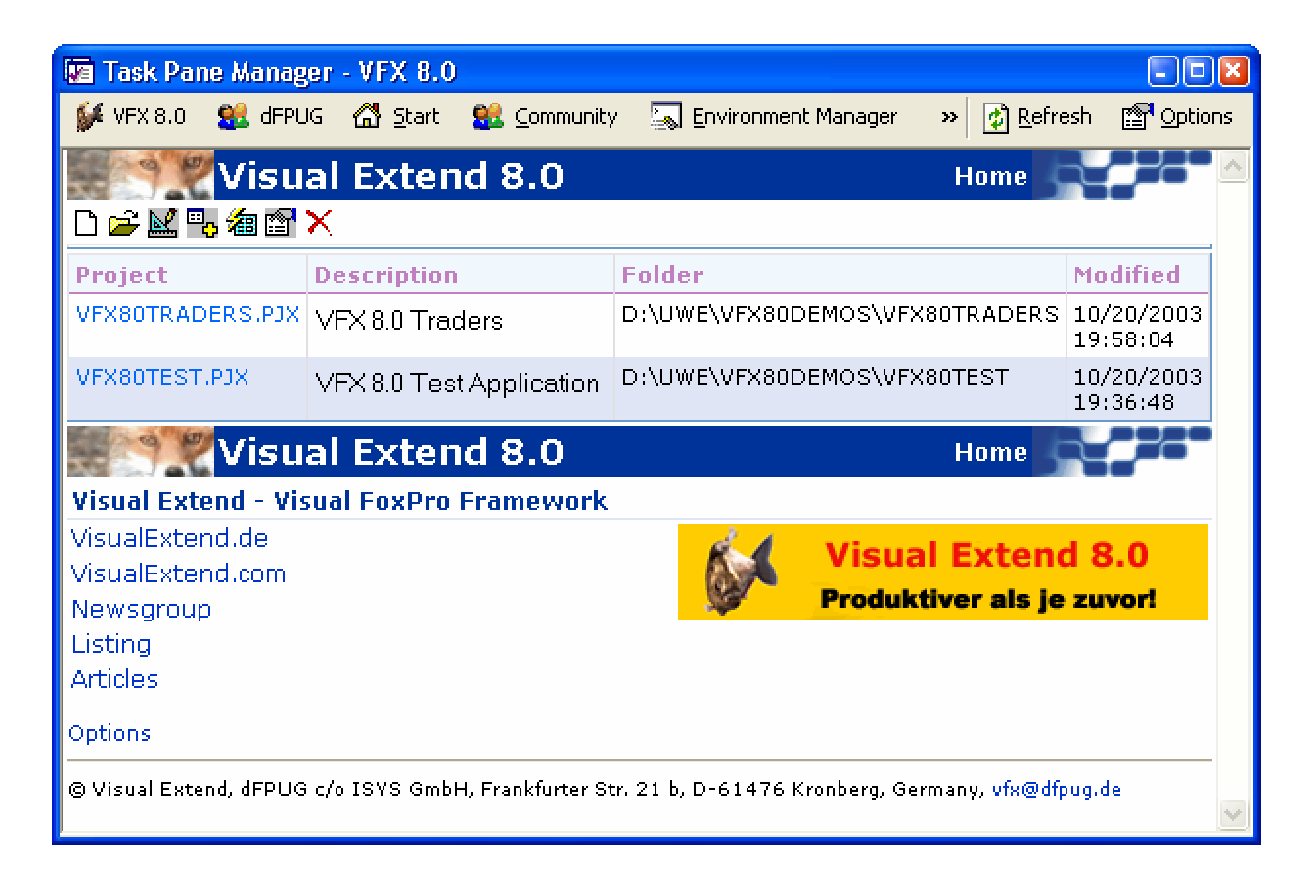
- Metadata Wizard

Der neue VFX-Menü-Designer:

- VMD (Visual Extend Menu Designer)

4.5. Die VFX 8.0 Task Pane

Der VFX – Application Manager wurde in die VFP Task Pane integriert.



Über die Symbolleiste stehen folgende Funktionen zur Verfügung:

New Project Startet den VFX – Application Wizard.

Open Project Öffnet ein VFP-Projekt und stellt den aktuellen Pfad auf den Projektordner.

Modify Project Öffnet das in der VFX 8.0 Task Pane selektierte Projekt und stellt den aktuellen Pfad auf den Projektordner.

Add Project Fügt ein vorhandenes VFP-Projekt der VFX 8.0 Task Pane hinzu.

Rebuild Neu kompilieren aller Dateien des in der VFX 8.0 Task Pane selektierten Projekts. Das Projekt wird nach dem kompilieren zur Bearbeitung geöffnet.

Properties Start der VFX – Project Properties zum in der VFX 8.0 Task Pane selektierten Projekt.

Delete Entfernt das selektierte Projekt aus der VFX 8.0 Task Pane.

5. Installation

5.1. Hardware- und Software-Anforderungen

Da es sich bei Visual Extend um eine Erweiterung zu Microsoft Visual FoxPro 8.0 handelt, benötigen Sie eine Hard- und Softwareumgebung, auf der Visual FoxPro 8.0 eingesetzt werden kann. Sehen Sie bitte bei den Systemanforderungen zu Microsoft Visual FoxPro nach.

5.2. Die Installation von VFX

Starten Sie das Installationsprogramm mit dem Namen *VFX80Setup.msi* und folgen Sie den Anweisungen auf dem Bildschirm.

Installieren Sie VFX 8.0 mit dem Installationsprogramm in einen neuen Ordner. Installieren Sie VFX 8.0 nicht in den Ordner, in dem sich eine frühere Version von VFX befindet!

VFX 8.0 hat einen Software-Kopierschutz. Nach der Installation, beim ersten Start eines VFX-Builders, zeigt Ihnen ein Dialog Ihre persönliche Registrierungsnummer. Sie brauchen Ihre Lizenz nur online auf unserer VFX-Registrierungswebseite zu registrieren und wir werden Ihnen den Aktivierungsschlüssel, den Sie in dem Dialog eingeben müssen, per E-Mail senden. Wir bieten zwei verschiedene Typen von Aktivierungsschlüsseln: Einer ist auf eine Laufzeit von 30 Tagen beschränkt, der andere ist unbefristet gültig.

Beachten Sie, dass Sie die Installation von VFX nicht von einem PC auf einen anderen PC kopieren können ohne einen neuen Aktivierungsschlüssel anfordern zu müssen. Ihre Registrierungsnummer wird aus den Daten Ihres PCs ermittelt und ist einmalig. Jeder VFX-Benutzer hat eine andere, einmalige Registrierungsnummer und muss sich daher online registrieren, um den Aktivierungsschlüssel zu bekommen. Erst dann ist die Arbeit mit den VFX-Buildern möglich. Der einzige Weg einen Aktivierungsschlüssel zu erhalten, ist die Registrierung auf unserer Webseite: http://www.visualextend.de

Heutzutage, wo sich die Dinge dramatisch schnell entwickeln, muss sich die große Investition, die VFX für uns und auch für Sie ist, in kurzer Zeit bezahlt machen. Ihre und unsere Investition muss bestmöglich geschützt werden.

Wir hoffen, dass Sie den neuen Softwarebasierten Schutz begrüßen und heißen Sie willkommen zur nächsten Generation von VFX. Dem besten VFX, das es je gab!

5.3. Einstellen der Visual FoxPro Umgebung für VFX

Sie müssen Microsoft Visual FoxPro 8.0 funktionsfähig installiert haben, bevor Sie die Arbeit mit VFX 8.0 beginnen können.

Als nächstes sollten Sie sicherstellen, dass das VFX 8.0-Menü jedes Mal automatisch erscheint, wenn Sie Ihr Visual FoxPro 8.0 starten. Wir schlagen folgenden Weg vor:

Fügen Sie diese Zeile der Datei **CONFIG.FPW** in Ihrem VFP 8.0-Ordner hinzu:

ANMERKUNG: Wenn Sie keine Datei mit dem Namen **CONFIG.FPW** haben, können Sie diese Datei mit dem Editor anlegen.

```
command = do (HOME() +"vfx.prg")
```

Diese Zeile teilt VFP mit, dass das Programm VFX.PRG ausgeführt werden soll, wenn VFP gestartet wird. In der Datei VFX.PRG (erstellen Sie diese Datei ebenfalls mit dem Editor und speichern Sie diese auch im VFP-Ordner) fügen Sie folgende Zeile hinzu:

```
do c:\programme\vfx80\builder\vfxmnu.app
```

Wir gehen hier davon aus, dass VFX im Ordner c:\Programme\VFX80 installiert ist. Passen Sie den Pfad ggf. an.

Beim Start des VFX-Menüs werden automatisch die folgenden Einstellungen in VFP gemacht:

- **Builder:** zeigt Sie auf den VFX-Anwendungs-Assistenten mit dem Namen *VFXBLDR.APP* im Ordner *\VFX80\BUILDER*.
- **Suchpfad:** *\VFX80\BUILDER* wird dem Suchpfad hinzugefügt.

Beim ersten Start von VFP nach der Installation von VFX 8.0 wird die VFX 8.0 Task Pane automatisch in die VFP Task Pane integriert.

Wichtiger Hinweis: Stellen Sie sicher, dass Sie sich immer im Ordner Ihrer Anwendung befinden! Benutzen Sie den Befehl *cd ?* im Befehlsfenster, um den aktuellen Pfad zu prüfen oder noch besser, verwenden Sie die VFX Task Pane für ein einfaches Wechseln zwischen den verschiedenen Projekten, ohne dass Sie den Ordner manuell ändern müssen. Wenn Sie sich in einem falschen Ordner befinden, wird Visual FoxPro unter Umständen andere Include-Dateien oder Klassenbibliotheken verwenden als Sie erwarten.

Das beste Werkzeug um zwischen Projekten zu wechseln, ist die VFX 8.0 Task Pane. Sie können die Task Pane über den VFP-Menüpunkt Extras, Task Pane öffnen. Wir empfehlen die VFP Task Pane beim Start von VFP automatisch öffnen zu lassen. Wählen Sie hierzu im Task Pane Manager die Option „Open the Task Pane Manager when Visual FoxPro starts".

5.4. Hinweis zur Einstellung der Klassenvorlagenzuordnung

Auf der Seite Formulare des Dialogs Extras/Optionen in VFP können Sie die Klassenvorlagen für Formularsätze (nicht von VFX unterstützt!) und Formulare einstellen. Wenn Sie die Klasse *CDataFormPage* als Klassenvorlage für neue Formulare eintragen, werden alle neuen Formulare automatisch auf Basis dieser Klasse erstellt. Aber seien Sie vorsichtig: **Das bedeutet, dass alle physikalischen Referenzen auf die eingestellte Klassenbibliothek und nicht auf Ihre projektspezifische Klassenbibliothek zeigen! Wir empfehlen daher, nicht mit den Klassenvorlagen für Formulare zu arbeiten. Benutzen Sie stattdessen den Formular-Assistenten.**

5.5. Hinweis zur Erstellung neuer Formulare mit dem VFX-Formular-Assistenten

Um ein neues Formular zu erstellen, ziehen Sie die Klasse *CDataFormPage* und lassen Sie diese auf das neue Formular fallen. Da Visual FoxPro bei diesem Vorgang automatisch ein *FormSet1* anlegt und das *Form1* erhalten bleibt, müssen Sie beides über das *Formular*-Menü löschen! **Wenn es Ihnen zu mühsam ist, neue Formulare auf diesem Weg zu erstellen, sollten Sie den VFX-Formular-Assistenten verwenden. Der Assistent wird über das VFX-Menü aufgerufen! Dieses Werkzeug ist wahrscheinlich der beste Grund, nicht die Klassenvorlagen zu verwenden, wie oben beschrieben.**

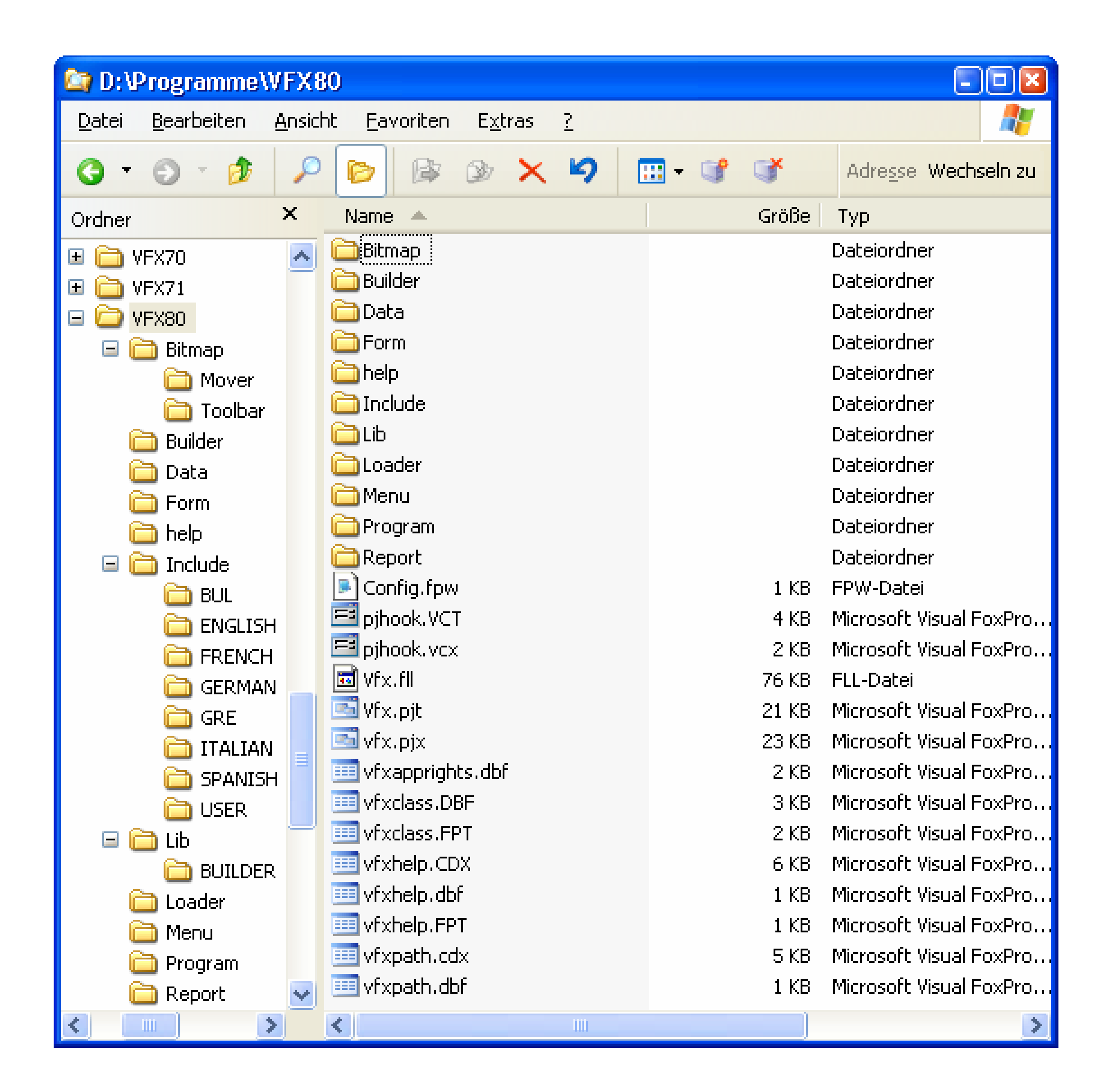
5.6. Hinweis für Entwickler mehrsprachiger Anwendungen

Um mehrsprachige Anwendungen zu entwickeln, müssen Sie im Ordner *\include* die Dateien der *gewünschten* Sprache haben. Alle VFX-Formulare haben bereits Code in der *LangSetup()*-Methode, sodass sie Bezeichnungen, Überschriften und Tooltips in der gewünschten Sprache anzeigen. In den VFX-

Klassenbibliotheken befinden sich keine sprachabhängigen Komponenten, sodass diese ohne Änderungen in allen Sprachen verwendet werden können.

5.7. Übersicht über die installierten Dateien

Nach der Installation von VFX haben Sie diese Ordnerstruktur im VFX-Ordner:



Der VFX-Ordner dient als zentraler Speicherplatz aller VFX-Komponenten und ist die Basis aller Projekte, die Sie mit dem VFX-Anwendungs-Assistenten erstellen (wie später in diesem Dokument beschrieben ist).

HINWEIS: Arbeiten Sie in diesem Projekt nicht direkt. Es ist NICHT für die direkte Bearbeitung gedacht. Verwenden Sie den Anwendungs-Assistenten, um ein neues Projekt zu erstellen, wie es später in diesem Dokument beschrieben ist.

6. Erstellen einer Anwendung mit dem VFX - Application Wizard

6.1. Ziel

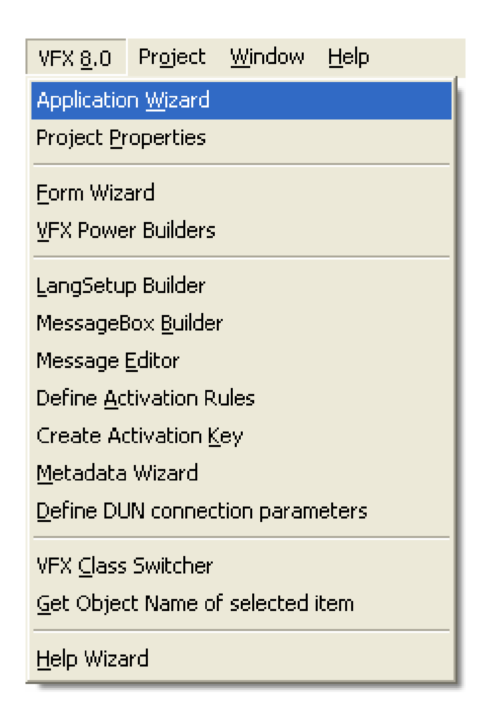
Wenn Sie ein neues Projekt beginnen, könnten Sie die ganze Ordnerstruktur von Hand erstellen, alle benötigten Dateien kopieren, wie etwa die Klassenbibliotheken, die Standardformulare, die Konfigurationsdateien, die Bilddateien usw. Hier greift der VFX-Anwendungs-Assistent ein: er erstellt das gesamte Projekt in der Sprache Ihrer Wahl. Er stellt außerdem die wichtigsten Eigenschaften der Anwendungsklasse ein und erstellt Include-Dateien mit den wichtigsten Konstanten, um die manuelle Arbeit so weit wie möglich zu reduzieren.

6.2. Vorbereitung

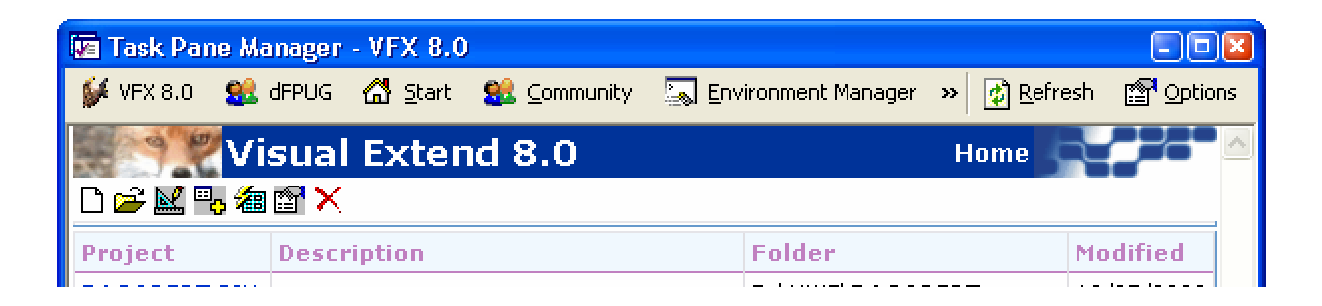
Schließen Sie alle Formulare und stellen Sie sicher, dass keine Klassenbibliotheken eines VFX-Projekts geöffnet sind. Am Besten beenden Sie Visual FoxPro und starten Sie erneut, bevor Sie den VFX-Anwendungs-Assistenten benutzen.

6.3. Der VFX - Application Wizard

Wählen Sie den Menüpunkt *Application Wizard* im VFX 8.0-Menü.



Oder starten Sie den *Application Wizard* aus der VFX Task Pane durch einen Klick auf das linke Symbol.



Der VFX-Application Wizard erscheint:

VFX - Application Wizard
1. With this wizard you create a new VFX project
Master VFX home folder: c:\programme\vfx80\
(Usually you don't need to modify this path.)
New Project
Enter the name of the new project file: MAIN
Enter the name of the new project's folder:
Database name: DATABASE.DBC
Click on next to proceed.
Cancel < Back Next > Finish

Geben Sie die folgenden Daten ein, bevor Sie eine neue Anwendung generieren lassen:

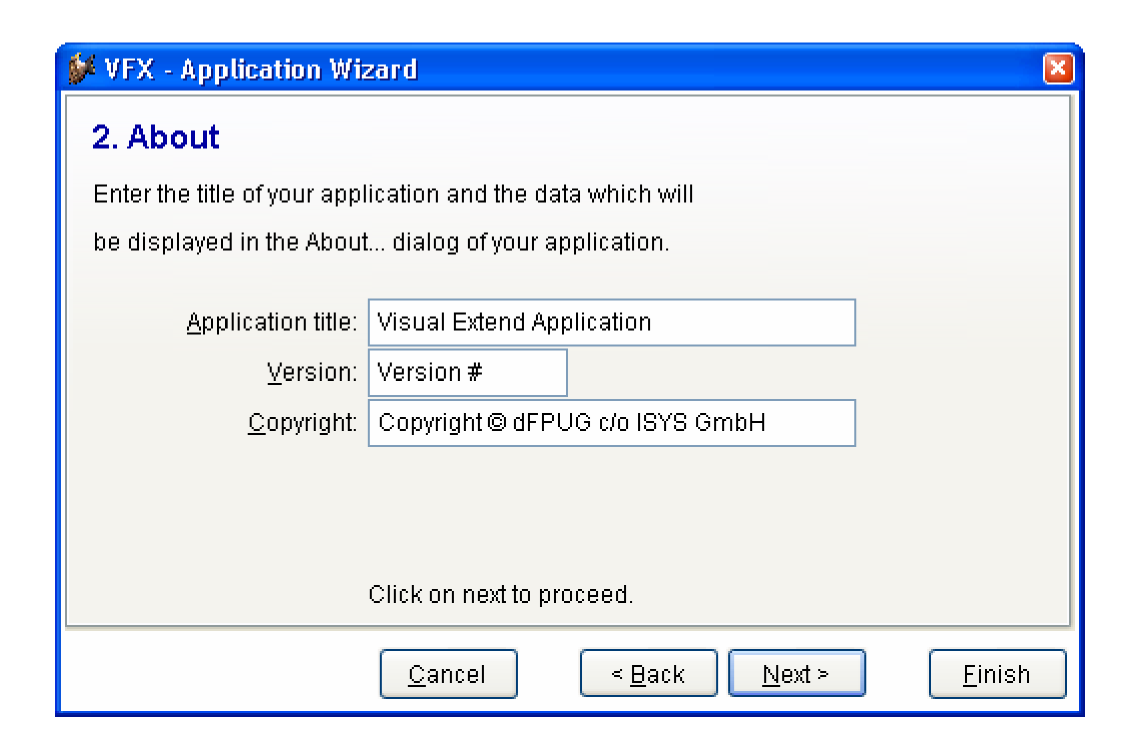
Master VFX home folder: Tragen Sie hier den VFX-Ordner ein, in dem sich Ihre VFX-Installation befindet. Normalerweise ist der vorgegebene Wert des Assistenten richtig und Sie brauchen keine Änderung zu machen.

Enter the name of the new project file: Geben Sie hier den Namen für Ihre neue Projektdatei ein. Fügen Sie keinen Pfad und keine Namenserweiterung hinzu. Geben Sie nur den Namen des neuen Projektes ein.

Enter the name of the new project's folder. Geben Sie den Ordner für Ihr neues Projekt ein. Wenn der Ordner noch nicht existiert, so wird er von dem VFX-Application Wizard erstellt.

Database Name. Geben Sie den Namen Ihres Datenbank-Containers an (DBC). Geben Sie nur den Namen des Datenbank-Containers ohne Pfad und ohne Namenserweiterung ein.

Auf der Seite mit dem Titel 2. About machen Sie die folgenden Eingaben:

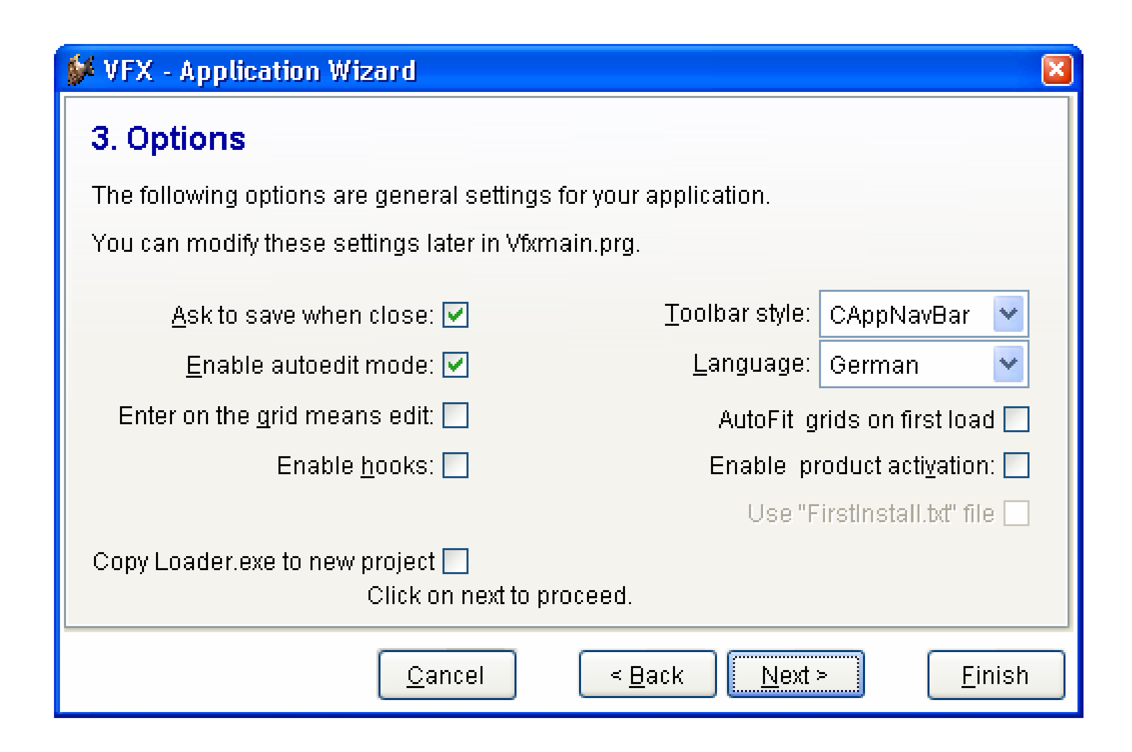


Application title: Geben Sie die Überschrift für das Hauptfenster Ihrer Anwendung an. Diese Überschrift wird als Konstante CAP_APPLICATION_TITLE in der Include-Datei USERTXT.H gespeichert.

Version: Geben Sie die Versionsnummer für den Infodialog Ihrer Anwendung ein. Die Nummer wird in der Konstante CAP_LBLVERSION in der Include-Datei USERTXT.H gespeichert.

Copyright: Geben Sie Ihre Copyright-Information für den Infodialog Ihrer Anwendung ein. Diese Information wird in der Konstante CAP_LBLCOPYRIGHTINFORMATION in der Include-Datei USERTXT.H gespeichert.

Auf der Seite mit dem Namen *3. Options* können Sie folgenden Optionen einstellen:



Ask to save when close: Die Auswahl dieser Option setzt den Wert der Eigenschaft *nAsktoSave* des Anwendungsobjekts auf 1. Diese Eigenschaft bestimmt das Verhalten von VFX wenn der Benutzer ein Formular schließt, nachdem er Änderungen am aktuellen Datensatz gemacht hat.

Enable autoedit mode. Die Auswahl dieser Option setzt den Wert der Eigenschaft *nAutoEditmode* des Anwendungsobjekts auf 1. Das bedeutet, dass der Benutzer jederzeit mit der Bearbeitung der Daten beginnen kann, ohne vorher in den Bearbeitungsmodus wechseln zu müssen.

Enter on the grid means edit: Die Auswahl dieser Option setzt den Wert der Eigenschaft *nEnterisEditinGrid* des Anwendungsobjekts auf 1. Das bedeutet, dass durch Drücken der Enter-Taste auf dem Grid einer Suchseite in den Bearbeitungsmodus gewechselt wird.

Enable hooks: Die Auswahl dieser Option setzt den Wert der Eigenschaft *nEnableHook* des Anwendungsobjekts auf 1. Das bedeutet, dass die Hooks aktiviert werden.

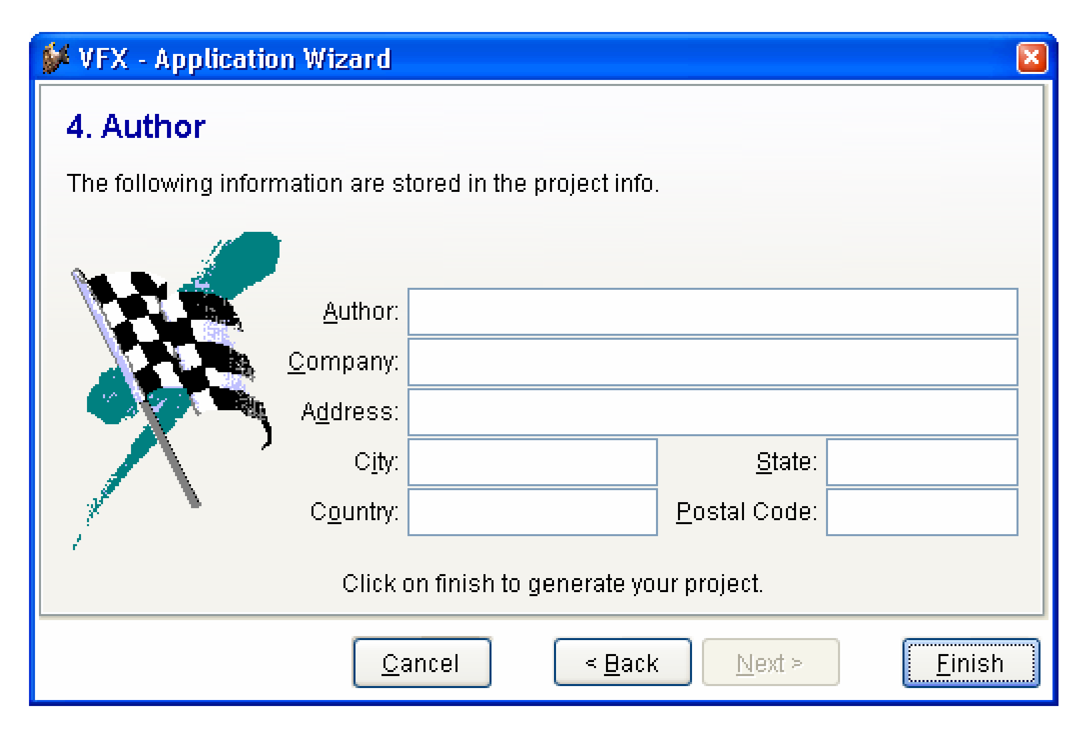
Toolbar style: Wählen Sie hier die Symbolleistenklasse, die Sie verwenden wollen. *CAppNavBar* enthält Schaltflächen zur Bewegung des Datensatzzeigers und andere Schaltflächen zur Bearbeitung in der Standard-Symbolleiste. *CAppToolbar* enthält keine Schaltflächen zur Bewegung des Datensatzzeigers und zur Bearbeitung.

Language: Wählen Sie die gewünschte Sprache für Ihr neues Projekt. Zurzeit können Sie aus folgenden Sprachen auswählen: deutsch, englisch, französisch, italienisch, spanisch, bulgarisch, tschechisch und griechisch.

Enable product activation: Die Auswahl dieser Option setzt den Wert der Eigenschaft *lUseActivation* des Anwendungsobjekts auf .T. Das bedeutet, dass die Applikation eine Produktaktivierung erfordert.

Use „Firstinstall.txt" file: Die Auswahl dieser Option setzt den Wert der Eigenschaft *lActivationType* des Anwendungsobjekts auf .T. Das bedeutet, dass die Produktaktivierung die Datei „Firstinstall.txt"erfordert. Der Schutz Ihrer Applikation wird dadurch weiter verbessert.

Auf der Seite *4. Author* können Sie Ihre persönlichen Daten eingeben, um Ihr Projekt zu dokumentieren.



Diese Informationen werden in der Projektdatei gespeichert.

6.4. Erstellen des Projekts

Wenn Sie *Finish* auswählen, wird der VFX-Application Wizard ein neues Projekt entsprechend den von Ihnen eingegebenen Parametern erstellen. Dabei wird die Musteranwendung aus der VFX-Installation in den neuen Projektordner kopiert. Die Include-Dateien werden entsprechend der ausgewählten Sprache kopiert. Anschließend wird das gesamte Projekt kompiliert, damit die in den Include-Dateien enthaltenen Konstanten zur Anwendung kommen.

Eine abschließende Meldung zeigt an, dass Ihre neue Applikation erfolgreich vorbereitet wurde.

ANMERKUNG: Da Sie sicher sofort mit der Arbeit an Ihrem neuen Projekt beginnen wollen, hat der VFX-Anwendungs-Assistent bereits automatisch den Standardordner auf den Startordner des neuen Projektes gesetzt. Um die Anwendung aus dem Projekt-Manager zu starten, wählen Sie das Hauptprogramm *VFXMAIN.PRG* und wählen Sie ausführen.

7. Diskussion der generierten VFX-Applikation

Nach einer erfolgreichen Anwendungsgenerierung mit dem VFX-Anwendungs-Assistenten, haben Sie eine lauffähige Anwendung mit allem was eine neue Anwendung benötigt vom Menü, über die Standard-Symbolleiste, die Benutzerverwaltung, die Systemeinstellungen, Datenbankwartung, ein Laufzeitfehlerprotokoll, ein Protokoll der Systemsperren bis hin zum Infodialog.

7.1. Office-kompatible Benutzeroberfläche

VFX erstellt Anwendungen, die nach dem *Office-Compatible*-Standard zertifiziert werden können.

7.1.1. Menü: Datei

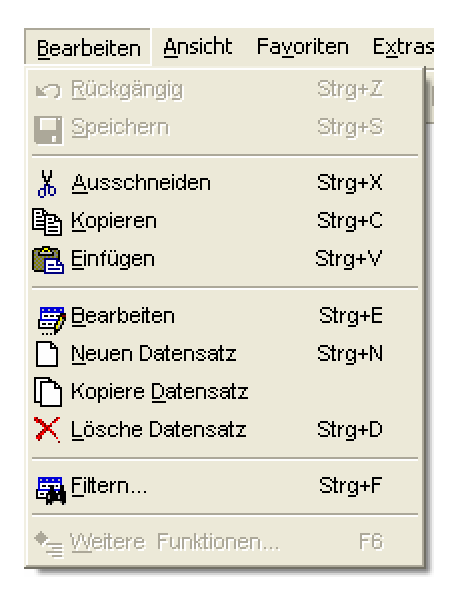


Mit einem Standard-*Datei/Öffnen*-Dialog wird die Komplexität von Menüs wesentlich reduziert. Der Benutzer öffnet Formulare immer durch einen einheitlichen Öffnen-Dialog. Standardmäßig wird der Öffnen-Dialog im Windows-XP-Stil am linken Bildschirmrand angezeigt.

VFX-Anwendungen bieten, dem *Office-Compatible*-Standard folgend, im Menü *Datei* die zuletzt geöffneten Dateien an. Wie viele Dateien angezeigt werden, ist für jeden Benutzer in der Benutzerverwaltung individuell einstellbar.

Auch die *Datei/Beenden* Option entspricht dem *Office-Compatible*-Standard.

7.1.2. Menü: Bearbeiten



Hier befinden sich alle Funktionen zur Datenbearbeitung, die sich auf den aktuellen Datensatz beziehen sowie die Möglichkeit, die Formulare für Filtern und weitere Funktionen aufzurufen. Je nach Status des Formulars

Bearbeitungs-/Einfügemodus (oForm.*nFormStatus* = 1 oder 2) oder

Anzeigemodus (oForm.*nFormStatus* = 0)

sind einige der Optionen nicht verfügbar.

Um weitere Informationen zu erhalten, sehen Sie bitte im Kapitel *Das VFX Datenbearbeitungsformular* nach.

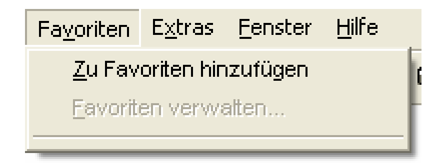
7.1.3. Menü: Ansicht



Hier können Sie den Symbolleisten-Dialog aufrufen, die Seite bei mehrseitigen Eingabeformularen wechseln, sowie den Datensatzzeiger bewegen.

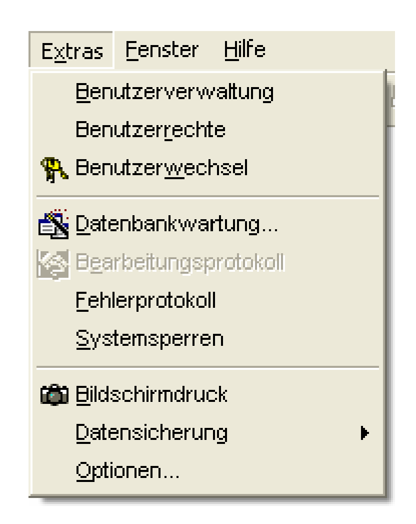
Um weitere Informationen zu erhalten, sehen Sie bitte im Kapitel *Das VFX Datenbearbeitungsformular* nach.

7.1.4. Menü: Favoriten



Dies ist das VFX-Favoriten-Menü. Mit der ersten Option wird der aktuelle Datensatz dem Favoriten-Menü hinzugefügt. Mit dem zweiten Eintrag werden die Favoriten verwaltet. Für alle verfügbaren Favoriten, gruppiert nach Formularen, werden Menüeinträge zur Laufzeit hinzugefügt.

7.1.5. Menü: Extras



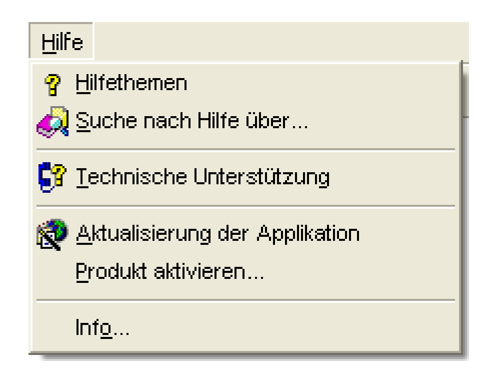
Um weitere Informationen zu den einzelnen Optionen zu erhalten, lesen Sie bitte in den Kapiteln Benutzerverwaltung, Benutzerrechte, Benutzerwechsel, Datenbankwartung, Bearbeitungsprotokoll und Fehlerprotokoll in diesem Handbuches nach.

7.1.6. Menü: Fenster



Falls Sie mehrere Fenster geöffnet haben, können Sie diese im Menü *Fenster* auswählen.

7.1.7. Menü: Hilfe



Das Hilfemenü bietet direkten Zugriff auf die Hilfedatei.

7.1.8. Standard-Symbolleiste

VFX-Anwendungen haben eine Standard-Symbolleiste, die Sie einfach um Ihre anwendungsspezifischen Schaltflächen erweitern können. Dadurch haben Benutzer einfachen Zugriff auf die Funktionen, die Ihre Anwendung bietet. Die VFX-Symbolleisten erscheinen im „Hot Tracking" Layout.

Standard

Neu (Strg+N)	Anlegen eines neuen Datensatzes.
Kopiere Datensatz	Der angezeigte Datensatz wird in einen neuen Datensatz kopiert.
Öffnen (Strg+O)	Öffnet den Öffnen-Dialog am linken Bildschirmrand.
Speichern (Strg+S)	Speichern der Änderungen im aktiven Formular.
E-Mail	Versenden einer E-Mail aus der Berichtsausgabe aus dem aktiven Formular.
PDF	Erstellen einer PDF-Datei aus der Berichtsausgabe aus dem aktiven Formular.
Drucken (Strg+P)	Drucken eines Berichts oder einer Liste aus dem aktiven Formular.
Seitenansicht	Anzeige der Druckvorschau eines Berichts oder einer Liste aus dem aktiven Formular.

Ausschneiden (Strg+X)	Entfernt die Markierung und überträgt sie in die Zwischenablage.
Kopieren (Strg+C)	Kopiert die Markierung in die Zwischenablage.
Einfügen (Strg+V)	Fügt den Inhalt der Zwischenablage ein.
Rückgängig (Strg+Z)	Macht die Änderungen in aktuellen Formular rückgängig.
Weitere Funktionen (F6)	Öffnet das Fenster mit weiteren Funktionen zum aktuellen Formular.
Bearbeitungsprotokoll	Öffnet das Formular mit dem Bearbeitungsprotokoll zum aktuellen Datensatz im aktiven Formular.
Bildschirminhalt drucken	Die aktuelle Bildschirmansicht wird gedruckt.
Bearbeiten (Strg+E)	Schaltet das aktive Formular in den Bearbeitungsmodus.
Löschen (Strg+D)	Löscht den aktuellen Datensatz im aktiven Formular.
Filtern (Strg+F)	Filtern der Daten im aktiven Formular nach einzugebenden Kriterien.
Anfang (Strg+Pos1)	Bewegt den Datensatzzeiger auf den Anfang der Tabelle oder Ansicht.
Rückwärts blättern (Strg+Pfeil oben)	Bewegt den Datensatzzeiger auf den vorherigen Datensatz der Tabelle oder Ansicht.
Vorwärts blättern (Strg+Pfeil unten)	Bewegt den Datensatzzeiger auf den nächsten Datensatz der Tabelle oder Ansicht.
Ende (Strg+Ende)	Bewegt den Datensatzzeiger auf das Ende der Tabelle oder Ansicht.
User	Beispiel für eine individuell zu verwendende Schaltfläche.
Hilfe (F1)	Aufruf der kontextsensitiven Hilfe.
Benutzerwechsel	Ermöglicht die Anmeldung eines anderen Benutzers während das Programm läuft.
Schließen (ESC)	Das aktive Formular wird geschlossen.

Neben dieser Standard-Symbolleiste bietet Ihnen VFX an, eine formularspezifische Symbolleiste zu definieren. Alles was Sie tun müssen, ist eine Symbolleisten-Klasse zu definieren und den Namen dieser Symbolleiste in der Formular-Eigenschaft *CToolbarClass* einzutragen. VFX erledigt alles Weitere für Sie automatisch.

HINWEIS: Für eine ausführliche technische Beschreibung zur Benutzung von formularspezifischen Symbolleisten lesen Sie bitte in der VFX Technischen Referenz nach.

7.1.9. Abschließende Bemerkung zur Office-Kompatibilität

Je nach Art Ihrer Anwendung kann es erforderlich sein, vom Office-Compatible-Standard abzuweichen. Das VFX-Menü zeigt eine Alternative, die die meisten Bedürfnisse, aber nicht alle, von möglichen Anwendungen abdeckt. Es lohnt sich einige Zeit in den Aufbau des Menüs und der Symbolleisten zu investieren, die Sie in Ihren Anwendungen verwenden wollen.

7.2. Datenbankwartung

Durch Auswahl des Menüpunktes *Extras, Datenbankwartung* erscheint der folgende Dialog:



In diesem Dialog sehen Sie eine Liste mit allen in Ihrer Anwendung verfügbaren Tabellen. In einem einfach zu bedienenden VFX-Mover-Dialog können die Tabellen ausgewählt werden, die bearbeitet werden sollen.

Es kann aus einer der folgenden Optionen ausgewählt werden:

- Komprimieren (pack)
- Memos packen (pack memo)
- Neu indizieren (reindex)

Drücken Sie nach der Auswahl auf *OK,* um die gewünschte Datenbankwartung durchzuführen.

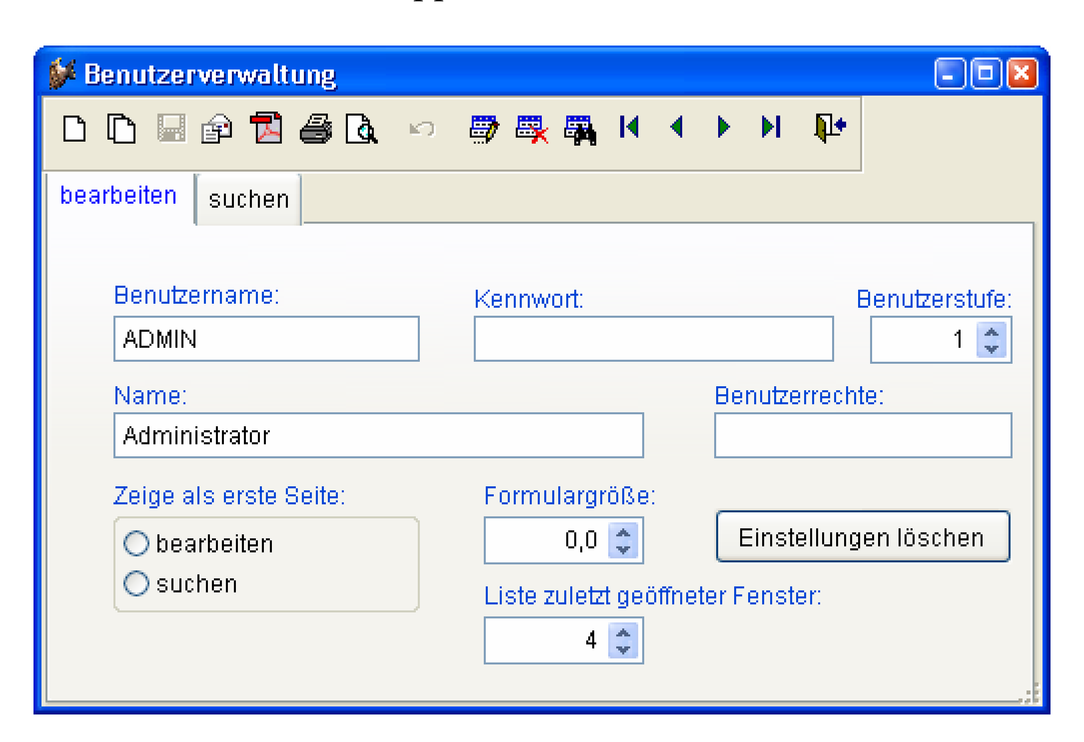
HINWEIS: Der hier verwendete Mover-Dialog ist ebenfalls eine VFX-Klasse und steht auch für Ihre eigenen Anwendungen zur Verfügung.

7.3. Benutzerverwaltung

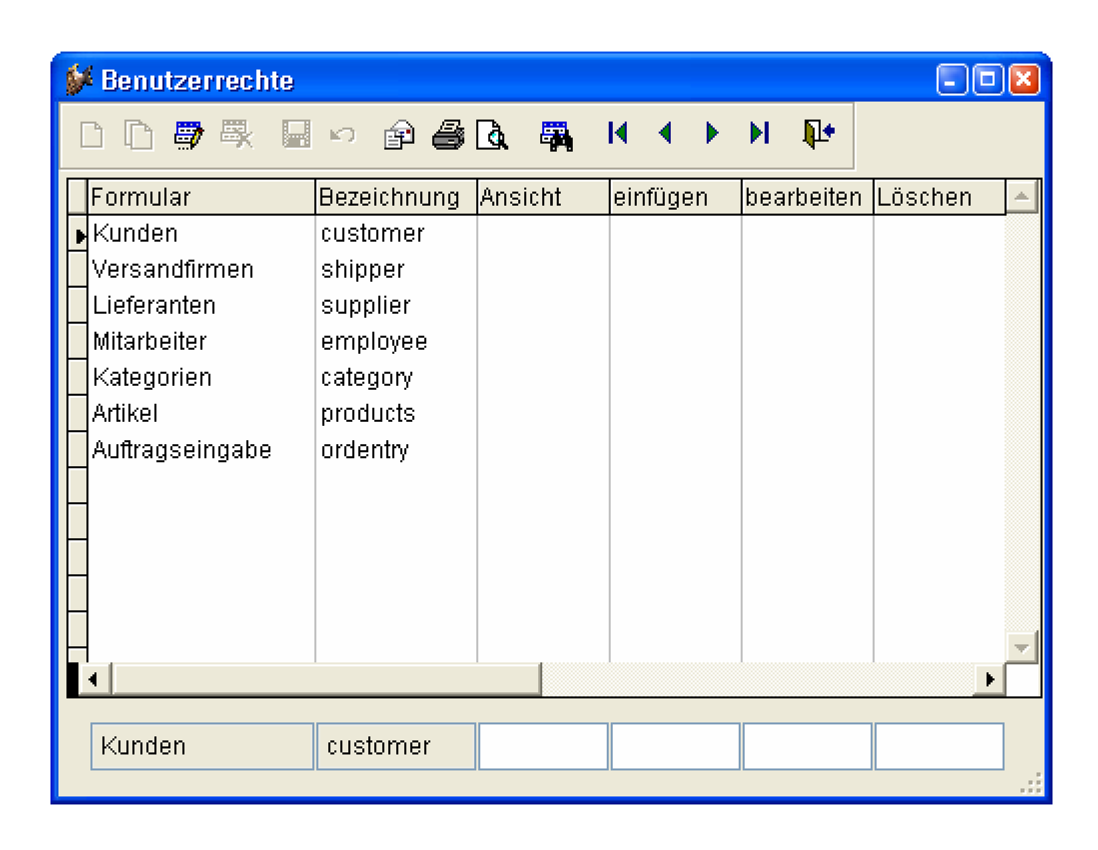
In jeder Mehrbenutzerapplikation sollte eine Benutzerverwaltung vorhanden sein. Als erstes muss festgelegt werden, wer zu Ihrer Anwendung Zugang hat. Dazu werden der Benutzername, das Kennwort und die Zugriffsrechte je Benutzer gespeichert. Eine weitere wichtige Funktion der Benutzerverwaltung ist die Speicherung der benutzerspezifischen Einstellungen.

Die Tabelle, in der die benutzerspezifischen Daten gespeichert sind, ist die freie Tabelle *VFXUSR.DBF/CDX*. Wenn Sie den Vorteil der langen Feldnamen nutzen möchten, können Sie diese Tabelle in Ihren Datenbank-Container einfügen.

Das Bearbeitungsformular, basierend auf der Klasse *CDataFormPage* wird automatisch vom VFX - Application Wizard vorbereitet.



Benutzer können ihre eigenen Daten in der VFX-Ressourcendatei löschen wenn sie mit neuen Einstellungen weitermachen wollen oder wenn sie von einer großen Bildschirmauflösung zu einer kleineren wechseln wollen oder wenn sie mit ihren bisherigen Einstellungen nicht mehr zufrieden sind. In der Ressourcendatei werden die Einstellungen für Formulargröße, Spaltenbreiten in Grids und Sortierfolgen in Grids und Auswahl-Grids sowie die Sortierfolgen gespeichert. Um die Daten in der VFX-Ressourcendatei zu löschen, drücken Sie auf die Schaltfläche *Einstellungen löschen*.



Die Zugriffsrechte werden über die Benutzerstufe gesteuert. Der Administrator hat die Benutzerstufe 1 und damit alle Rechte. Ein Benutzer, der die Benutzerstufe 99 hat, hat die wenigsten Rechte. Im Formular Benutzerrechte kann für jedes Formular festgelegt werden welche Benutzerstufe erforderlich ist um das Formular anzeigen zu können, um neue Datensätze erfassen zu können, um vorhandenen Datensätze bearbeiten zu können und um Datensätze löschen zu können.

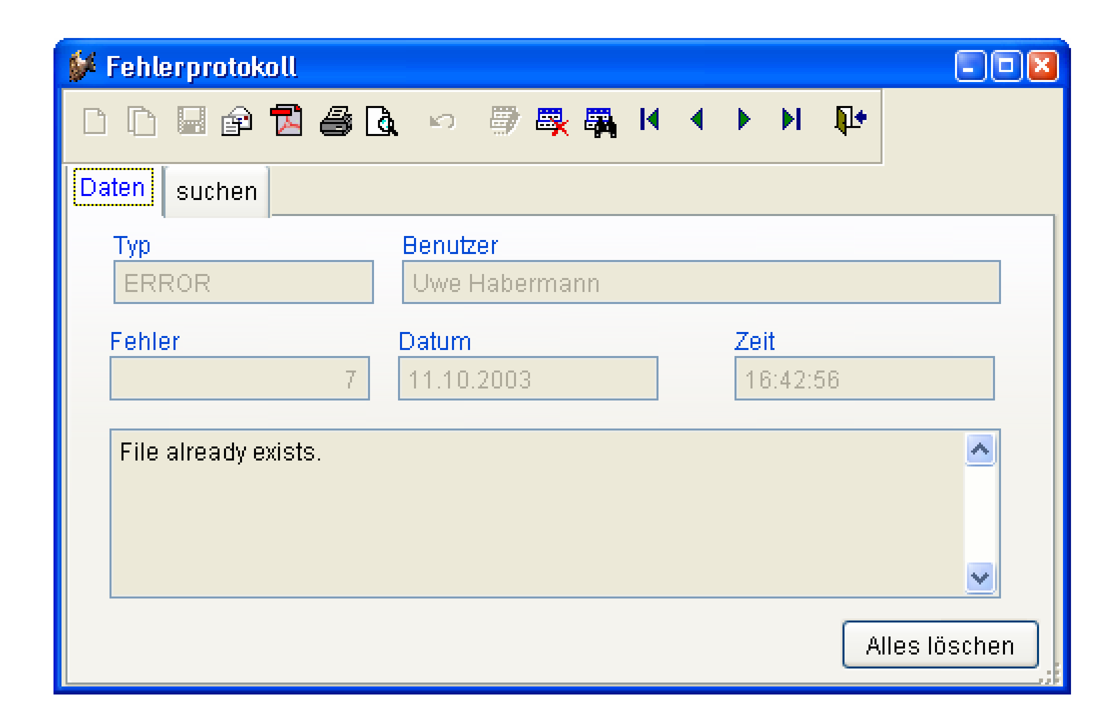
ANMERKUNG: Benutzer können nicht die Daten von anderen Benutzern ändern, wenn diese eine höhere Sicherheitsstufe haben. Sicherheitsstufen starten mit 1 (Administrator) und enden mit 99 (niedrigste Sicherheitsstufe). Zusätzlich können Sie eine Zugriffszeichenfolge für die weitere Anpassung an Ihre Bedürfnisse festlegen. Für weitere Sicherheitsaspekte, besonders für alle VFX Formular-Sicherheitseigenschaften, lesen Sie bitte in der VFX Technischen Referenz nach.

Wenn ein Benutzer nicht das Recht hat ein Formular anzuzeigen, wird das betreffende Formular nicht instanziiert. Solange im Dialog Benutzerrechte keine Benutzerstufen eingetragen sind, gelten die Einstellungen, die mit dem VFX - Form Wizard in den Formular-Eigenschaften *lcaninsert*, *lcancopy*, *lcanedit* und *lcandelete* hinterlegt sind.

7.4. Fehlerprotokoll

VFX protokolliert alle Laufzeitfehler automatisch. Die Tabelle mit den Fehlermeldungen ist die freie Tabelle *VFXLOG.DBF/CDX*.

Das Bearbeitungsformular, basierend auf der Klasse *CDataFormPage*, wird automatisch vom VFX Anwendungs-Assistenten vorbereitet.



Der Administrator kann das Fehlerprotokoll mit der Schaltfläche *Alles löschen* löschen.

ANMERKUNG: Für weitere Informationen lesen Sie bitte in der VFX Technischen Referenz nach.

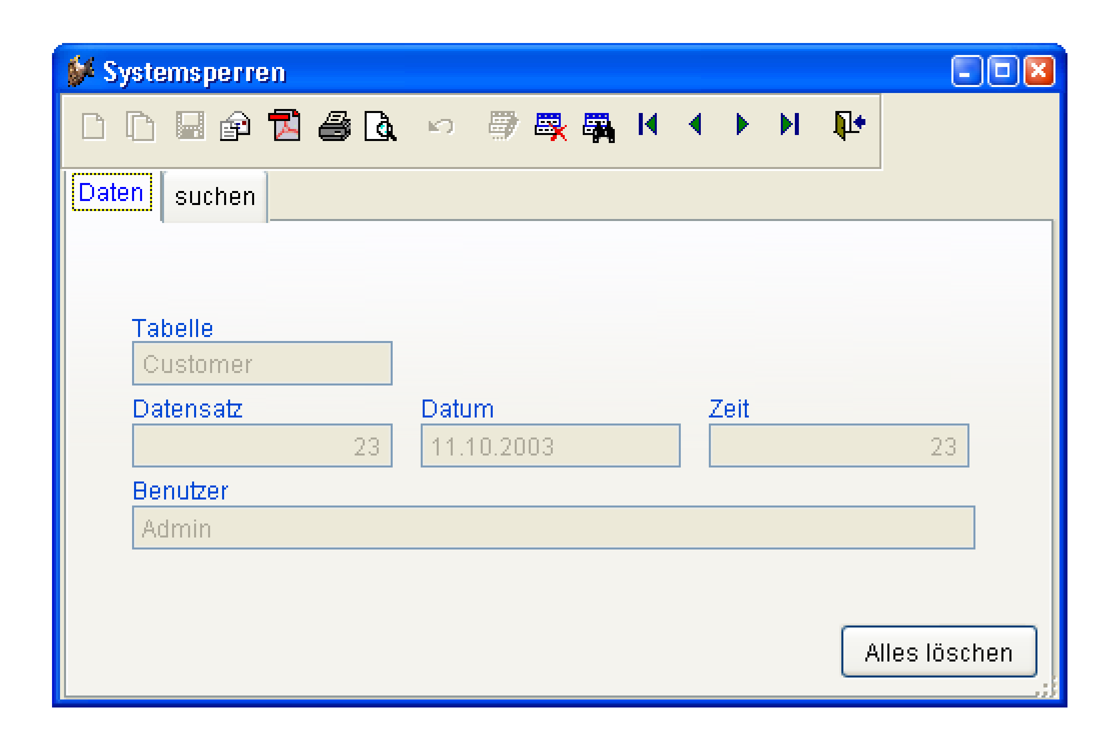
7.5. Systemsperren

In viel benutzten Mehrbenutzerumgebungen kann eine Meldung wie *"Datensatz durch anderen Benutzer gesperrt"* unter Umständen nicht ausreichen. Für solche Fälle stellt VFX eine System-Sperrentabelle zur Verfügung. In dieser Tabelle wird gespeichert, welcher Benutzer seit wann welchen oder welche Da-

tensätze in Benutzung hat. (Siehe die Funktionen *XLock()* sowie *XUnlock()* in der Technischen Referenz unter *Funktionen*).

Die Systemsperrentabelle in der alle Sperren mit VFX-Funktionsaufrufen gespeichert werden, ist die freie Tabelle *VFXLOCK.DBF/CDX*.

Das Bearbeitungsformular basiert auf der VFX-Klasse *CDataFormPage* und wird automatisch durch den VFX Anwendungs-Assistenten vorbereitet.

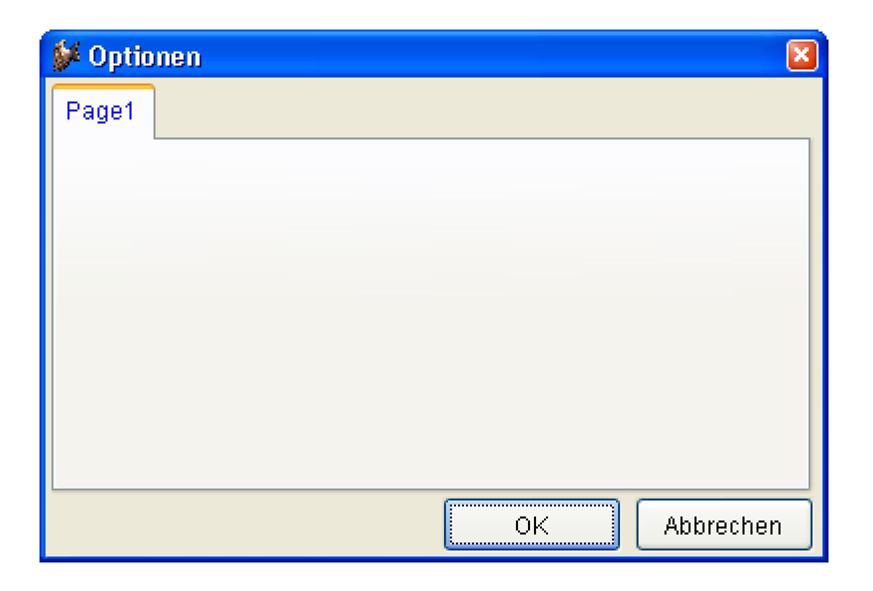


Der Administrator kann die Systemsperren mit der Schaltfläche *Alles löschen* löschen.

ANMERKUNG: Für weitere Informationen lesen Sie bitte in der VFX Technischen Referenz nach.

7.6. Optionen

Im Gegensatz zu den benutzerspezifischen Einstellungen werden in der Tabelle *VFXSYS.DBF* die systemspezifischen Einstellungen gespeichert.



Das oben abgebildete Formular ist eine Vorlage, die für die eigenen Optionen verwendet werden kann.

Der VFX Anwendungs-Assistent erstellt das Formular *VFXSYS* für Sie in einer gebrauchsfertigen Form. Dieses Formular basiert auf der Klasse *CSystemDialog*. Alles was Sie noch tun müssen ist, die gewünschten Felder in der *VFXSYS* Tabelle anzulegen und die entsprechenden Steuerelemente direkt mit der jeweiligen Public Variablen *gs_* als Datenquelle einzufügen.

Auch hier, wie bei den benutzerspezifischen Einstellungen, die weiter oben beschrieben sind, wird für jedes Feld aus der Tabelle *VFXSYS* eine Public Variable mit dem Präfix *gs_* angelegt. VFX übernimmt auch hier vollautomatisch das Speichern und Wiederherstellen dieser Werte falls diese aus dem Optionen-Dialog heraus verändert werden.

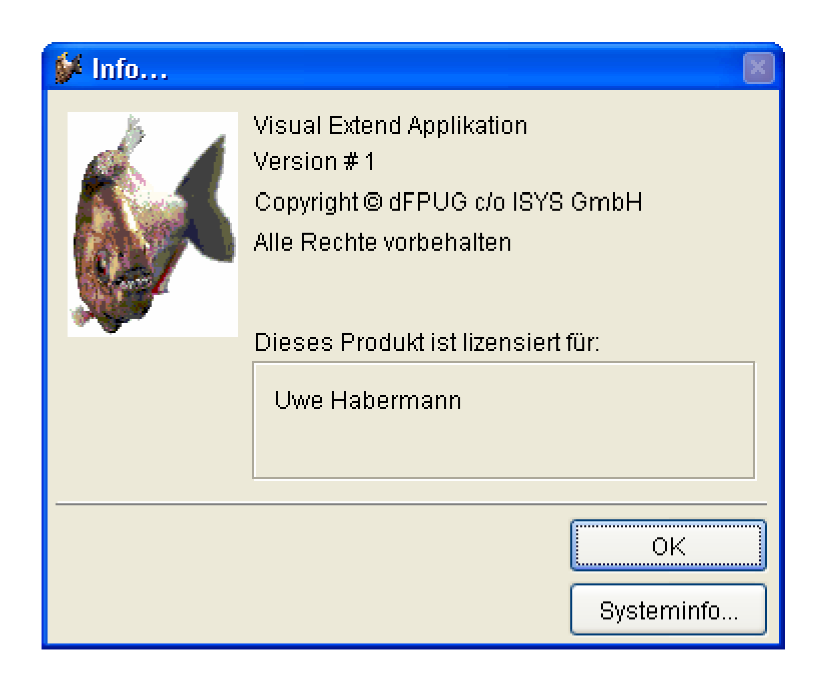
Wenn Sie ein Feld mit dem Namen *TEST* in der Tabelle *VFXSYS* haben, wird eine Public Variable *gs_test* den Wert aus dem Feld *TEST* der *VFXSYS*-Tabelle beinhalten. Falls diese Variable verändert wird, wird beim Verlassen des Optionen-Dialogs dieser Wert wieder zurück in das Feld *Test* der Tabelle *VFXSYS* geschrieben.

Auf diese Weise ist es sehr einfach, systemspezifische Einstellungen zu speichern und wiederherzustellen. Es genügt hierzu, in der Tabelle *VFXSYS* die entsprechenden Felder als Variable hinzuzufügen. Das ist sehr einfach. Probieren Sie es!

7.7. Infodialog

Der VFX-Anwendungs-Assistent erstellt einen Infodialog, der auf der Klasse *CAboutDialog* basiert.

Sie finden den Infodialog im Menü Hilfe.



Um diesen Dialog Ihren Bedürfnissen anzupassen, steht Ihnen die Include-Datei *USERTXT.H* zur Verfügung:

```
…
#define CAP_APPLICATION_TITLE              "VFX 8.00 Build 0000 Test Application"
#define CAP_LBLCOPYRIGHTINFORMATION              "Copyright © dFPUG c/o ISYS GmbH"
#define CAP_LBLTHISPRODUCTISLICENSEDTO           "This product is licensed to:"
#define CAP_LBLTRADEMARKINFORMATION              "Trademark Information"
#define CAP_LBLVERSION                           "Version "
#define CAP_LBLYOURAPPLICATIONNAME               "VFX Test Application"
…
```

HINWEIS: Wenn Sie Änderungen in dieser Include-Datei machen, müssen Sie das Formular *VFXABOUT.SCX* vor dem Start Ihrer Anwendung öffnen und speichern. Sonst werden die Änderungen in der Include-Datei nicht übernommen.

8. Die VFX-Builder

8.1. Ziel

Formulare manuell zu erstellen kann viel Zeit beanspruchen, insbesondere dann, wenn Sie viele Formulare mit vielen Feldern anzeigen möchten. Stellt man sich zum Beispiel ein Formular mit 20 Feldern vor, so hat man bereits 40 Steuerelemente, allein für die Dateneingabefelder: 20 Textfelder oder andere Steuerelemente und 20 Bezeichnungen. Wenn Sie Klassenbibliotheken verwenden, müssen Sie Ihre Symbolleiste anpassen oder die gewünschten Steuerelemente per drag & drop auf das Formular ziehen. Mit den VFX-Formular-Buildern wird diese Aufgabe sehr schnell und einfach durchführbar.

Ein weiterer großer Vorteil der VFX-Formular-Builder ist die Widerverwendbarkeit. Das bedeutet, dass Sie Änderungen, die Sie in Ihrer Datenbank gemacht haben, einfach in das bestehende Formular übernehmen können, indem Sie den VFX-Formular-Builder aufrufen und das Kontrollkästchen *Use DBC Definitions* auswählen. Auch Seiten zu einem Seitenrahmen hinzuzufügen oder Änderungen an den Spalten eines Grids sind sehr einfach dank der Widerverwendbarkeit der VFX-Formular-Builder.

8.2. Ergebnis

Bitte lesen Sie im Abschnitt *Diskussion des VFX Standard-Datenbearbeitungsformulars* später in diesem Kapitel, um eine Vorstellung von der Bedienung eines von VFX erzeugten Standard-Bearbeitungsformulars zu bekommen.

8.3. Vorbereitung

8.3.1. Erstellen der Datenbank

Zuerst müssen Sie die Datenbank für Ihre Anwendung erstellen. Legen Sie Ihre Tabellen, Felder und Indexschlüssel an.

ANMERKUNG: Wenn Sie die Daten für Überschriften, Formate, Eingabeformulare und Bibliothek für Anzeige im Datenbank-Container speichern, werden diese automatisch von den VFX-Formular-Buildern und vom VFX-Grid-Builder verwendet.

8.3.2. Erstellen eines neuen Formulars

Starten Sie den *VFX – Form Wizard* aus dem VFX-Menü. Geben Sie dem Formular einen Namen und wählen Sie die Klasse, auf der das Formular basieren soll. Es ist möglich aus einer der VFX-Formularklassen auszuwählen oder eine eigene Formularklasse zu verwenden. Eigenen Formularklassen müssen Ableitungen aus einer VFX-Formularklassen sein. Das generierte Formular wird im *Form*-Ordner unterhalb des aktuellen Projekts gespeichert und sofort im Formular-Designer geöffnet.

8.3.3. Einrichten der Datenumgebung

Richten Sie die Datenumgebung für das zu erstellende Formular ein. Die VFX-Formular Builder holen sich automatisch die Informationen aus der Datenumgebung für den Erstellungsprozess.

8.4. Der VFX-CDataFormPage Builder

8.4.1. Aufruf eines VFX-Formular-Builders

Um einen VFX-Formular-Builder aufzurufen, bewegen Sie die Maus auf den weißen Hintergrund des Formular-Designers, drücken Sie die rechte Maustaste und wählen Sie Builder.

ANMERKUNG: Wenn Sie eine Meldung erhalten, dass es keinen Builder für das ausgewählte Objekt gibt oder wenn ein Standard-Visual FoxPro-Builder erscheint, prüfen Sie, ob Sie das Formularobjekt ausgewählt haben. **Ein verbreiteter Fehler ist, den Seitenrahmen anstelle des Formularobjektes auszuwählen. Prüfen Sie das angezeigte Objekt im Eigenschaftsfenster, wenn Sie nicht sicher sind, ob das Formularobjekt ausgewählt ist.**

Der VFX-CDataFormPage Builder wird geladen und zeigt einen benutzerfreundlichen Dialog:

8.4.2. Die Bedienung des VFX-Formular-Builders

Die VFX-Formular-Builder haben eine intuitive Bedienung.

Form Name. Geben Sie den Namen des neuen Formulars ein. Der VFX – Form Wizard hat bereits einen Standardnamen entsprechend den Namenskonventionen zugewiesen. Der Name beginnt mit *frm*. Selbstverständlich können Sie Ihrem Formular einen beliebigen Namen geben, aber wir empfehlen Ihnen, sich an die allgemeinen Namenskonventionen zu halten.

Caption. Geben Sie die Überschrift für Ihr Formular ein. Während Sie die Überschrift eingeben, wird diese bereits in der Überschrift des Formular-Builders angezeigt. Wenn Ihr Formular veränderliche Überschriften in Abhängigkeit vom Aufruf des Formulars haben soll, brauchen Sie sich um diese Überschrift keine Gedanken zu machen. Geben Sie in diesem Fall einfach eine mehr oder weniger zutreffende Überschrift ein.

Der VFX – CDataFormPage Builder hat einen Seitenrahmen mit drei Seiten mit den Namen *Edit Pages*, *Grid Page* und *Form Options*. Auf der Seite *Edit Pages* definieren Sie den Seitenrahmen, den Sie zur Bearbeitung der ausgewählten Felder benutzen. Auf der *Grid Page* definieren Sie das *Grid* für die Suche und auf der Seite *Form Options* setzen Sie verschiedene Optionen für das Formular.

Die folgenden Optionen sind auf der Seite *Edit Pages* verfügbar:

Page Count. Geben Sie ein, wie viel Bearbeitungsseiten Sie benötigen. Für einige Formulare wird eine Bearbeitungsseite genug sein. Wenn Sie mehr Fel-

der haben, werden Sie diese auf mehrere Seiten verteilen wollen. In Abhängigkeit von der Anzahl der gewählten Seiten, sehen Sie im Seitenrahmen des Formular-Builder einen Seitenrahmen, der diese Seiten anzeigt. Wenn Sie zwei Bearbeitungsseiten eingeben, sehen Sie zwei Seiten auf dem Seitenrahmen, wenn Sie drei Bearbeitungsseiten eingeben, sehen Sie drei Seiten auf dem Seitenrahmen usw.

Page Title. Geben Sie die Überschrift der aktuellen Bearbeitungsseite ein. Wenn Sie die Überschrift für die zweite Seite eingeben wollen, drücken Sie auf die zweite Seite und Sie können die Überschrift auch für diese Seite eingeben. Der VFX-Formular-Builder zeigt während der Eingabe die sich ergebende Überschrift für die einzelnen Seiten an.

Justified Tab. Markieren Sie dieses Kontrollkästchen, wenn die Seitenüberschriften justiert sein sollen. Ansonsten haben die Überschriften eine variable Länge und füllen nicht die Breite des Seitenrahmens.

Für jede Bearbeitungsseite stehen die folgenden Optionen zur Verfügung:

Fields Selected. Hier sehen Sie alle Felder, die Sie für die aktuelle Bearbeitungsseite ausgewählt haben. Um Felder hinzuzufügen benutzen Sie das *Field Assistant*-Fenster, das in einem eigenen Formular angezeigt wird und alle aus der Datenumgebung zur Verfügung stehenden Felder anzeigt.

Control Type. Geben Sie für alle ausgewählten Felder den zu benutzenden Steuerungstyp an. Die folgenden Klassen stehen hierfür zur Verfügung:

Steuerelement	Beschreibung	VFX Klassenbibliothek
<Standardwert>	**Die Klasse, die Sie als Klasse für die Anzeige im Datenbank-Container angegeben haben (Standardwert).**	
CTextBox	Normales Textfeld.	VFXOBJ.VCX
CKeyField	Textfeld für das Bearbeiten von Identifikationsfeldern, die nach Anlegen des Datensatzes nicht mehr verändert werden dürfen.	VFXOBJ.VCX
CFixField	Textfeld für das Bearbeiten von Feldern, die in einer Child-Tabelle mit einer Haupttabelle verbunden sind. Dieses Steuerelement wird verwendet, wo ein Child-Formular von einem Parent-Formular aufgerufen wird und einen festen Wert aus dem Parent-Formular übergeben erhält, z. B. bei Aufträgen von einem Kunden. In diesem Fall würde das Kundenfeld im Formular Aufträge ein *CfixField* sein, denn im Falle des Aufrufes der Aufträge von einem Kunden muss das Kundenfeld vorbelegt werden und darf nicht änderbar sein.	VFXOBJ.VCX
CPickField	Eingabefeld, bei dem die Eingabe mit Hilfe einer Tabelle oder Ansicht überprüft wird und auf Wunsch eine Auswahlliste zur Verfügung gestellt wird. Es wird nicht nur die Eingabe überprüft, sondern zusätzlich ein beliebiger Ausdruck aus der Prüftabelle geholt.	VFXOBJ.VCX
CEditBox	Bearbeitungsfeld für das Bearbeiten von Memofeldern und längeren Zeichenfeldern.	VFXOBJ.VCX
CComboBox	Kombinationsfeld.	VFXOBJ.VCX
CListBox	Listenfeld.	VFXOBJ.VCX
CCheckBox	Kontrollkästchen für logische Felder.	VFXOBJ.VCX
COptionGroup	Optionsgruppe.	VFXOBJ.VCX
CSpinner	Drehfeld für numerische Felder.	VFXOBJ.VCX

ANMERKUNG: Um Ihre eigenen Klassen zu verwenden, tragen Sie diese im Datenbank-Container bei jedem Feld bei „Bibliothek für Anzeige" ein!

Caption. Überschrift für das ausgewählte Feld. Der Standardwert wird aus dem Datenbank-Container übernommen.

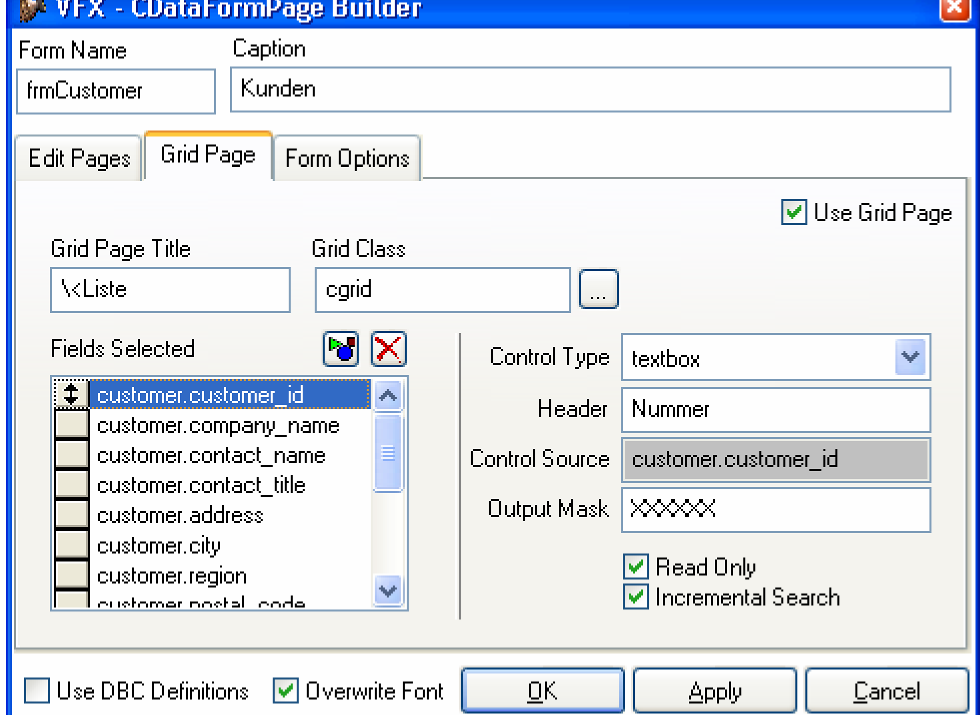
Format. Format-Eigenschaft für das selektierte Feld. Der Standardwert wird aus dem Datenbank-Container übernommen.

Input Mask. Eingabemasken-Eigenschaft für das selektierte Feld. Der Standardwert wird aus dem Datenbank-Container übernommen.

Status Bar. Meldung für die Statuszeile für dieses Feld. Der Standardwert wird aus dem Datenbank-Container übernommen. (Eigenschaft Feldkommentar, wenn dieser Wert leer ist, wird die Feldüberschrift genommen).

Read only. Wenn ein Steuerelement nur zur Anzeige von Informationen verwendet wird, markieren Sie dieses Kontrollkästchen.

Die folgenden Optionen stehen auf der Seite *Grid Page* zur Verfügung:



Use Grid Page. Markieren Sie dieses Kontrollkästchen, wenn Sie eine Listenseite auf Ihrem Formular haben wollen.

Grid Page Title. Geben Sie die Überschrift für die letzte Seite Ihres Formulars ein, die normalerweise ein Grid mit allen Datensätzen Ihrer Tabelle oder Ansicht enthält.

Grid Class. Geben Sie die Klasse für das Grid ein oder benutzen Sie den Standardwert, die *CGrid*–Klasse.

Fields Selected. Hier sehen Sie alle für das Grid ausgewählten Felder. Um Felder auszuwählen, benutzen Sie das *Field Assistant*-Fenster, in dem alle Felder aus der Datenumgebung zur Auswahl stehen.

Calculated Fields. Drücken Sie auf diese Schaltfläche um ein beliebiges berechnetes Feld hinzuzufügen.

Control Type. Geben Sie für alle ausgewählten Felder den gewünschten Kontrolltyp an. Die folgenden Kontrolltypen sind verfügbar (aus Gründen der Geschwindigkeitsoptimierung bieten wir nur VFP-Basisklassen für das Grid an):

Kontrolltyp	Beschreibung	VFP-Basisklasse
Textbox	Textfeld (Standard)	TEXTBOX
Editbox	Bearbeitungsfeld	EDITBOX
Combobox	Kombinationsfeld	COMBOBOX
Checkbox	Kontrollkästchen	CHECKBOX

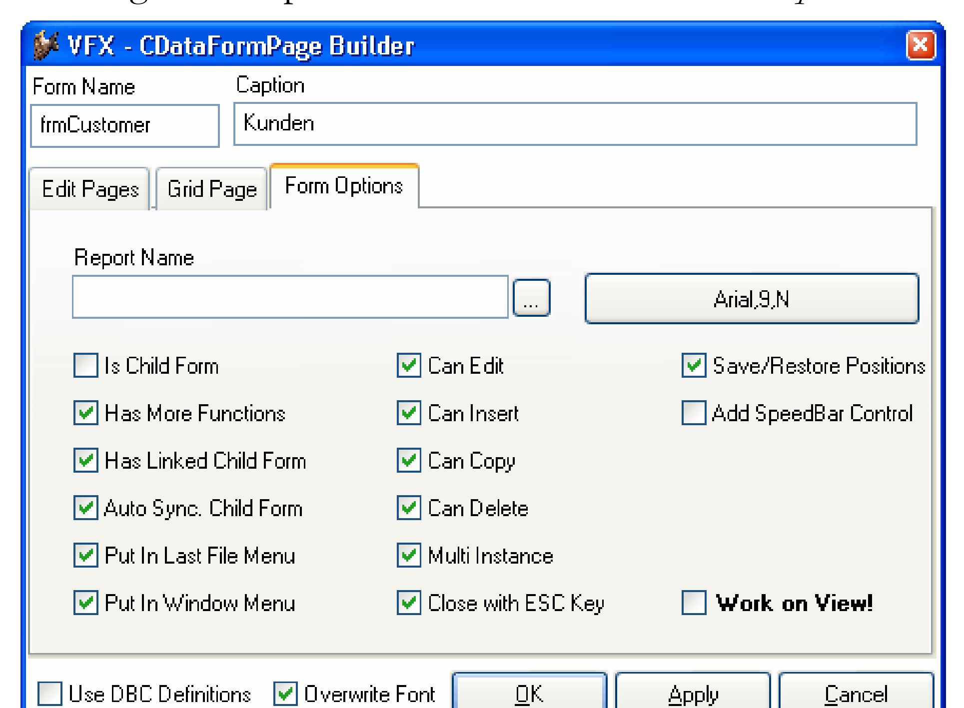
Header. Überschriften für die Spalten Ihres Grids. Die VFX Formular-Builder fügen automatisch die Überschriften aus dem Datenbank-Container ein.

Output Mask. Die VFX-Formular-Builder erstellen die Ausgabemaske anhand der Feldlänge. Sie können die Ausgabemaske ändern, um sie an Ihre Bedürfnisse anzupassen.

Read only. Wenn ein Steuerelement nur zur Anzeige von Informationen verwendet wird, markieren Sie dieses Kontrollkästchen.

Incremental Search. Markieren Sie dieses Kontrollkästchen, wenn Sie die inkrementelle Suche für die ausgewählte Spalte aktivieren wollen. Beachten Sie, dass VFX eine temporäre Indexdatei erstellt, wenn kein Indexschlüssel für die Spalte vorhanden ist. (Mit der *CGrid*-Eigenschaft *nMaxRec* können Sie angeben ab welcher Anzahl Datensätze dem Benutzer eine Meldung angezeigt werden soll, bevor eine temporäre Indexdatei erstellt wird.)

Die folgenden Optionen sind auf der Seite *Form Options* verfügbar:



Report Name. Hier können Sie den Namen eines Berichts eingeben. Wenn der Benutzer *drucken* oder *Seitenansicht* wählt, wird dieser Bericht gedruckt bzw. angezeigt. Sie brauchen für diese Funktionalität keinen Code in die Methode *OnPrint()* einzufügen. Wenn diese Eigenschaft leer gelassen wird, sucht VFX nach einem Bericht, der gleichen Namen wie das Formular hat.

Is Child Form. Wenn das Formular, das Sie gerade erstellen, von einem anderen Formular aufgerufen wird, ist dieses Formular ein Child-Formular.

ANMERKUNG: Bitte verwechseln Sie dies nicht mit dem später beschriebenen 1:n-Formular, wo Sie die Haupttabelle und die Child-Tabelle **auf dem gleichen Formular** bearbeiten können. Hier sprechen wir über folgendes Verhalten: Formular 1 ruft Formular 2 auf, wobei Formular 1 das Hauptformular und Formular 2 das Child-Formular ist. Im Formular 2 sehen Sie nur die Datensätze, die ein bestimmtes Kriterium erfüllen, das die Verbindung zur Haupttabelle im Formular 1 herstellt.

Wenn Sie beispielsweise in einem Formular die Aufträge eines Kunden anzeigen wollen, markieren Sie dieses Kontrollkästchen und der VFX-Formular-Builder wird das Formular automatisch als Child-Formular erstellen. Dabei werden automatisch die erforderlichen Codezeilen in die *Init()*-Methode des Formulars eingetragen. Sie müssen nur noch den Code der *Init()*-Methode prüfen und an Ihre Bedürfnisse anpassen.

Für weitere Details lesen Sie bitte im Abschnitt *Erweiterte Formulareigenschaften mit dem VFX-Formular-Builder* weiter unten in diesem Handbuch nach.

ANMERKUNG: Wenn Sie ein Formular haben, das sowohl als Child-Formular als auch als normales Formular dienen soll, markieren Sie die Option *Is Child Form*. Sie brauchen hierfür nicht zwei Formulare zu erstellen. Ein Formular kann sowohl alle Aufträge darstellen als auch nur die Aufträge eines bestimmten Kunden.

Has More Functions. Wenn das Formular, das Sie gerade erstellen, andere Formulare aufrufen oder Aktionen ausführen soll, müssen Sie dieses Kontrollkästchen markieren. Dadurch wird automatisch der erforderliche Code für die *OnMore()*-Methode Ihres Formulars erstellt. Sie müssen nur noch den Code in der *OnMore()*-Methode an Ihre Bedürfnisse anpassen. Normalerweise werden Sie eine Anzahl von Aktionen haben, die zur Auswahl in einem Formular angeboten werden. Der Benutzer kann dann die gewünschte Aktion auswählen.

Für weitere Details lesen Sie bitte im Abschnitt *Erweiterte Formulareigenschaften mit dem VFX-Formular-Builder* weiter unten in diesem Handbuch nach.

Has Linked Child Form. Wenn das Formular, das Sie gerade erstellen, Child-Formulare aufrufen soll, die dynamisch mit diesem Hauptformular verbunden bleiben, markieren Sie dieses Kontrollkästchen. Dadurch wird automatisch der Code für die Formularmethode *OnSetChilddata()* erstellt. Diese Methode wird automatisch für jedes vorhandene Child-Formular aufgerufen.

Autosynch Child Form. Hiermit wird die Formulareigenschaft *lAutosynchChildform* festgelegt. Dadurch wird angegeben, ob die Child-Formulare automatisch mit diesem Hauptformular synchronisiert werden, wenn Sie den Datensatzzeiger im Hauptformular bewegen.

Put in Last File Menu. Hiermit wird die Formulareigenschaft *lPutinLastFile* festgelegt. Sie gibt an, ob die Formularüberschrift in die Liste der benutzen Dateien im Menü *Datei* eingetragen werden soll.

Put in Window Menu. Hiermit wird die Formulareigenschaft *lPutinWindowmenu* festgelegt. Sie gibt an ob das laufende Formular in das Menü *Fenster* eingetragen werden soll. Beachten Sie auch die Eigenschaft *nWinMnuCount* und die Methode *RefreshWindowMenu()* im Anwendungsobjekt.

Can Edit. Hiermit wird die Formulareigenschaft *lCanEdit* festgelegt. Sie gibt an, ob der Benutzer Datensätze im aktuellen Formular bearbeiten kann.

Can Insert. Hiermit wird die Formulareigenschaft *lCanInsert* festgelegt. Sie gibt an, ob der Benutzer Datensätze im aktuellen Formular einfügen kann.

Can Copy. Hiermit wird die Formulareigenschaft *lCanCopy* festgelegt. Sie gibt an, ob der Benutzer Datensätze im aktuellen Formular kopieren kann.

Can Delete. Hiermit wird die Formulareigenschaft *lCanDelete* festgelegt. Sie gibt an, ob der Benutzer Datensätze im aktuellen Formular löschen kann.

Multi Instance. Hiermit wird die Formulareigenschaft *lMultiInstance* eingestellt. Standardmäßig können alle Formulare, die Sie mit VFX erstellen, mehrmals geöffnet werden (das nennt man multiinstanzfähig). Dies ist eine großartige Eigenschaft. Alles was Sie dabei beachten müssen ist, dass das Formular mit einer privaten Datensitzung arbeiten muss. Das ist der Standardwert in allen VFX-Formularen.

Trotzdem ist es manchmal günstig, die Eigenschaft multiinstanzfähig ausschalten zu können. Daher haben wir die Eigenschaft *lMultiInstance* eingeführt. Setzen Sie diese Eigenschaft auf *.F.* und das Formular kann nur einmal geöffnet werden.

Close with ESC key. Hier wird die Formulareigenschaft *lCloseonEsc* eingestellt, die angibt, ob der Benutzer ein Formular mit der Escape-Taste schließen kann.

Save/Restore positions. Hier wird die Formulareigenschaft *lSavePosition* eingestellt, die angibt, ob die Positionen und andere Formulareinstellungen in der VFX-Ressourcendatei gespeichert werden sollen.

Add Speedbar Control. Dieses Kontrollkästchen fügt dem Formular eine Schaltflächenleiste hinzu. Hier ein Beispiel:



OK. Wählen Sie diese Schaltfläche, um Ihr Formular generieren zu lassen. Dies dauert einige Sekunden und das Ergebnis ist ein Formular, auf dem Sie die gewünschte Anzahl von Bearbeitungsseiten mit den gewählten Feldern auf jeder Seite haben. Wenn Sie mehr Felder gewählt haben als untereinander auf eine Seite passen, werden zwei Spalten erzeugt.

Der Formularerstellungsprozess kann mehrmals gestartet werden. Diese Eigenschaft nennt man wieder verwendbar.

Anmerkung: Die Eigenschaft wieder verwendbar ist zu 100% nur für Formulare verfügbar, die mit dem VFX-Formular-Builder erzeugt wurden. Um das wieder verwendbare Verhalten des Builders sicherzustellen sollten Sie immer den VFX-Formular-Builder verwenden, wenn Sie Ihrem Formular Felder hinzufügen wollen.

Ein weiterer großer Vorteil der wieder verwendbaren VFX-Formular-Builder ist die Tatsache, dass Sie Änderungen, die Sie in der Datenbank (z. B. Über-

schrift, Format oder Eingabemaske) durchgeführt haben, durch Aufrufen des VFX-Formular-Builders und auswählen des Kontrollkästchens *Use DBC Definitions* in das Formular übernehmen können.

Apply. Hat die gleiche Funktion wie die Schaltfläche *OK*, schließt den VFX-Formular-Builder jedoch nicht.

Cancel. Bricht die Ausführung des VFX-Formular-Builders ab. Jede Auswahl und Eingabe geht dabei verloren.

8.5. Der VFX – Cgrid Builder

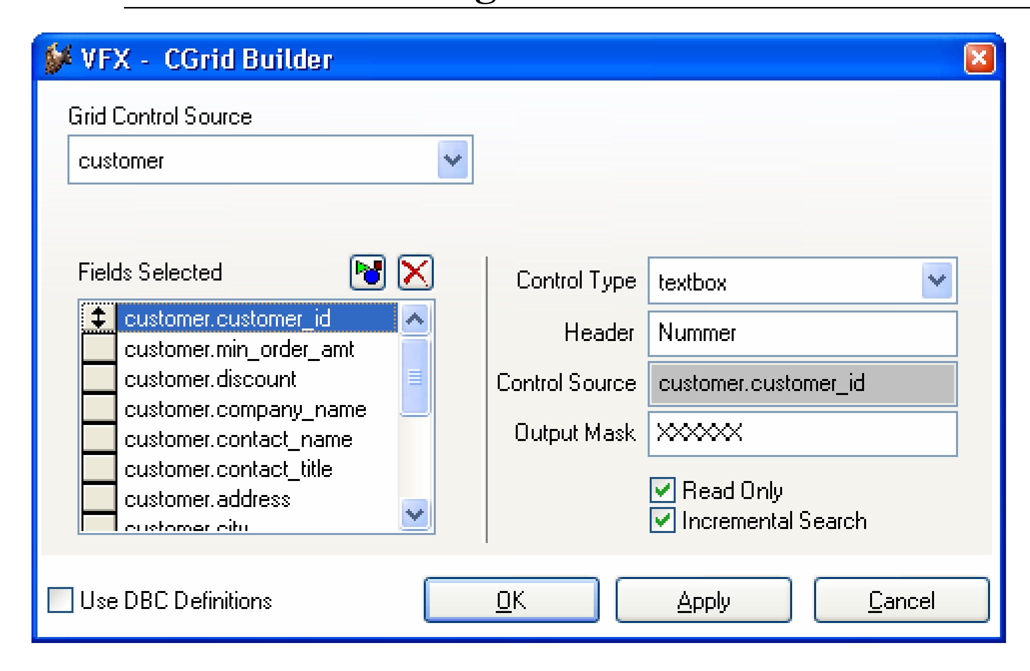
8.5.1. Aufruf des VFX – Cgrid Builder

Obwohl der VFX-Formular-Builder bereits eine Seite mit einem Grid anlegt, kann es sein, dass Sie nur in diesem Grid Änderungen durchführen wollen. Der VFX - CGrid Builder automatisiert die Erstellung von leistungsfähigen Grids. Die resultierenden VFX Power Grids sind einfach zu bedienen und bringen keine Geschwindigkeitseinbußen mit sich. Sie werden die Eigenschaften der VFX Power Grids sehr nützlich finden. Die inkrementelle Suche sowie die benutzerspezifische Speicherung der Spaltenreihenfolge, Spaltenbreiten und Sortierfolge des Grids werden von den Benutzern Ihrer Anwendung geschätzt werden.

Um den VFX – CGrid Builder aufzurufen, wählen Sie die letzte Seite Ihres Formulars und wählen Sie das Grid-Steuerelement aus. Um den Builder aufzurufen, drücken Sie die rechte Maustaste und wählen Sie Builder.

Der VFX – CGrid Builder wird geladen und zeigt den folgenden Dialog:

8.5.2. Die Bedienung des VFX-Grid-Builder



Die Bedienung ist die gleiche wie auf der Grid-Seite des VFX-Formular-Builders. Für eine detaillierte Beschreibung aller Optionen lesen Sie bitte die Beschreibungen im Abschnitt *Der VFX-Formular-Builder* nach.

8.6. Der VFX-Pickfield Builder

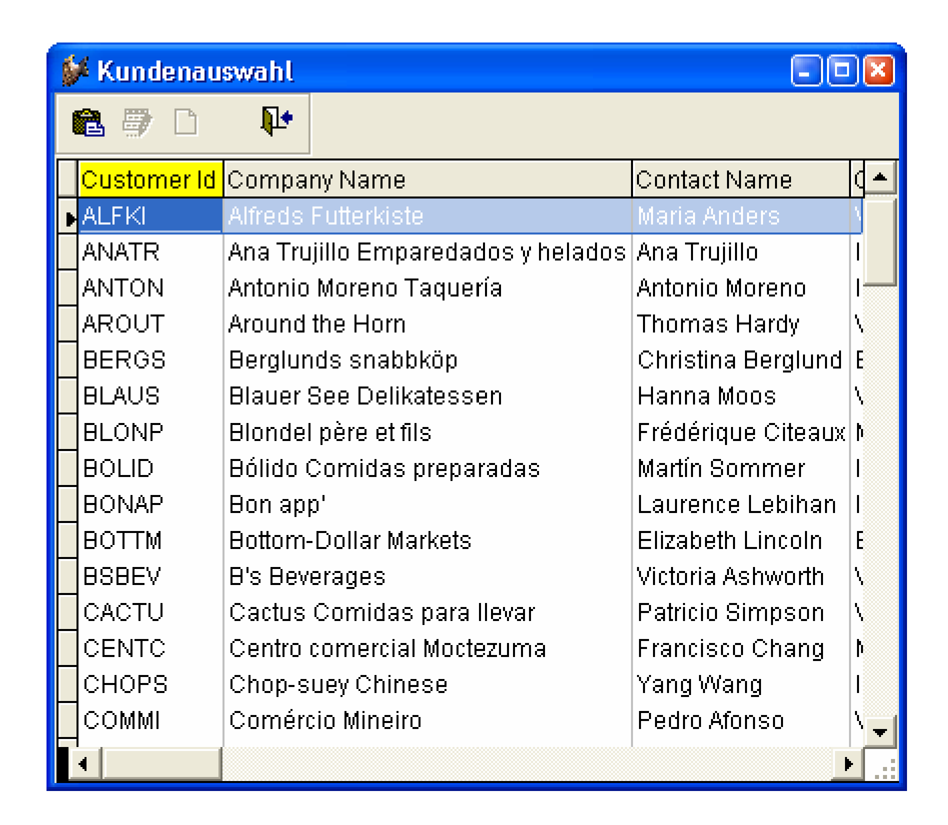
8.6.1. Ergebnis

Wenn Sie ein Auswahllisten-Steuerelement auf einem Formular einsetzen, sieht das etwa so aus:



Der Benutzer kann die Auswahlliste auf folgende Weise aufrufen:

- Drücken der Schaltfläche neben dem Auswahllisten-Eingabefeld (normalerweise mit drei Punkten beschriftet).
- Doppelklick auf das Auswahllisten-Eingabefeld oder auf den Beschreibungstext.
- Drücken der Funktionstaste F9.



Der Dialog der Auswahlliste hat folgende Eigenschaften (wie jedes VFX Power Grid):

- Inkrementelle Suche mit automatischer Einstellung der Sortierfolge.
- Einstellen der Sortierfolge durch Doppelklick auf die Spaltenüberschrift.
- Die Breite der Spalten kann verändert werden.
- Position und Gestaltung des Grids werden automatisch gespeichert.

Der Benutzer kann den gewünschten Datensatz auf folgende Weise auswählen:

- Doppelklick.
- Drücken der Taste Eingabetaste.
- Drücken der Schaltfläche Übernehmen.

Wenn der Benutzer die Tabelle bearbeiten möchte, die der Auswahlliste zugrunde liegt, kann er auf die Schaltfläche *Bearbeiten…* drücken. Daraufhin erscheint das Bearbeitungsformular für diese Tabelle. Wenn der Benutzer neue Datensätze hinzufügen will, drückt er auf die Schaltfläche *neu*.

8.6.2. Aufruf des VFX – CPickField Builder

Ein Steuerelement, das Sie häufig benutzen werden, ist die Auswahlliste. Dieses Container-Steuerelement bietet Ihnen eine einfache Möglichkeit Textfelder hinzuzufügen, die die Benutzereingabe mit Werten aus einer Tabelle oder Ansicht überprüfen. Außerdem kann der Benutzer ein Auswahllisten-Formular öffnen, um den gewünschten Datensatz auszuwählen. Da die Auswahllisten-Klasse einige Eigenschaften hat, die definiert werden müssen, hilft Ihnen der VFX – CPickField Builder bei der einfachen Erstellung der Auswahllisten-Steuerelemente. Und das ohne eine einzige Zeile Code oder Text im Eigenschaftsfenster des Auswahllisten-Containers manuell eintragen zu müssen!

Um den VFX – CPickField Builder aufzurufen, wählen Sie das Auswahllisten-Container-Steuerelement auf dem Formular, drücken die rechte Maustaste und wählen Builder.

ANMERKUNG: Um ein Steuerelement auszuwählen, das sich auf einer Seite in einem Seitenrahmen auf einem Formular befindet, müssen Sie den Visual FoxPro-Weg benutzen, um Steuerelemente innerhalb der Containerhierarchie auszuwählen (Klick, Rechtsklick, bearbeiten). Eine gute Möglichkeit, um festzustellen ob Sie das richtige Steuerelement ausgewählt haben, ist ein Blick in das Eigenschaftsfenster.

Der VFX – CPickField Builder wird geladen und zeigt den folgenden Dialog:

8.6.3. Die Bedienung des VFX – CPickField Builder

VFX - CPickField Builder
Pick Field | Update | Work on View | Options
Pick Dialog Caption: Kundenauswahl
Maintenence Form:
Pick Table Name: customer
Pick Table Index Tag: customer_i
CPickField::txtField.ControlSource: orders.customer_id
CPickField::txtDesc.ControlSource: customer.company_name
Return Field Name (Code).Use STR() for Num. Fields.: customer_id
Return Field Name (Description): company_name
Format:
Input Mask: XXXXXX
Status Bar Text: Kund/innen-Nr. des Bestellers
OK | Apply | Cancel

Auch dieser Builder ist voll wieder verwendbar. Das bedeutet, dass Sie diesen Builder während des Entwicklungsprozesses beliebig oft verwenden können ohne die Eigenschaften zu verlieren, die Sie bereits eingestellt haben.

Auf der Seite *Pick Field* stehen die folgenden Optionen zur Verfügung:

Pick Dialog Caption. Geben Sie die Überschrift für das Auswahllisten-Formular ein. In diesem Formular kann der Benutzer einen Wert auswählen.

Maintenance Form. Wenn der Benutzer den gewünschten Datensatz in dem Auswahllisten-Formular nicht findet, möchten Sie dem Benutzer vielleicht die Möglichkeit geben, das normale Bearbeitungsformular (im Ansichtsmodus oder gleich im Einfügemodus) aufzurufen. Geben Sie hier den Namen für das Bearbeitungsformular ein. Es wird aufgerufen, wenn der Benutzer auf die Schaltfläche *Bearbeiten…* im Auswahllisten-Formular drückt.

Pick Table Name. Wählen Sie den Namen der Tabelle oder Ansicht aus der Sie den Wert auswählen oder überprüfen möchten. Hier können Sie zwischen allen Tabellen oder Ansichten aus der Datenumgebung wählen.

Pick Table Index Tag. Dieser Indexschlüssel wird zur Überprüfung der Benutzereingabe verwendet.

CPickField::txtField.ControlSource. Dies ist die Datenquelle für das Eingabetextfeld.

CPickField::txtDesc.ControlSource. Wählen Sie die Datenquelle für das Beschreibungsfeld des Auswahllisten-Steuerelementes. Stellen Sie sicher, dass Sie eine korrekte Beziehung zu der Tabelle herstellen aus der diese Datenquelle stammt. Andernfalls wird dieses Steuerelement nicht den gewünschten Wert anzeigen, wenn Sie den Datensatzzeiger in Ihrem Formular bewegen.

Return Field Name (Code). Geben Sie den Namen des Feldes (aus der Tabelle oder Ansicht der Auswahlliste) ein, das den ausgewählten Wert enthält. Geben Sie keinen Aliasnamen ein, weil Tabellen für Auswahllisten mit einem temporären Namen geöffnet werden.

Return Field Name (Description). Geben Sie den Namen des Feldes (aus der Tabelle oder Ansicht der Auswahlliste) ein, das den Wert mit der Beschreibung enthält. Geben Sie keinen Aliasnamen ein, weil Tabellen für Auswahllisten mit einem temporären Namen geöffnet werden.

Format. Der VFX – CPickField Builder übernimmt diese Eigenschaft aus dem Datenbank-Container.

Input Mask. Der VFX – CPickField Builder übernimmt diese Eigenschaft aus dem Datenbank-Container.

Status Bar Text. Der VFX – CPickField Builder übernimmt diese Eigenschaft aus dem Datenbank-Container.

OK. Die eingestellten Optionen werden in das ausgewählte Auswahllisten-Objekt eingefügt.

Apply. Macht das gleiche wie *OK,* jedoch wird der VFX – CPickField Builder nicht beendet.

Cancel. Bricht die Arbeit mit dem VFX – CPickField Builder ab. Alle Eingaben werden verworfen.

Auf der Seite *Update* stehen die folgenden Optionen zur Verfügung:

VFX - CPickField Builder
Pick Field | Update | Work on View | Options
Update Source Fields
company_name;address;city;region;postal_code;country
Target Table Name
orders
Update Target Fields
ship_to_name;ship_to_address;ship_to_city;ship_to_region;ship_to_postal_code;ship_to_country
OK | Apply | Cancel

Update Source Fields. Hier können sie Felder aus der Auswahlliste eingeben, deren Werte in die Bearbeitungstabelle übernommen werden sollen. Wenn Sie mehrere Werte eingeben, so müssen diese durch Semikolon getrennt werden.

Target Table Name. Wählen sie die Zieltabelle aus. Normalerweise ist dies die Bearbeitungstabelle des Formulars.

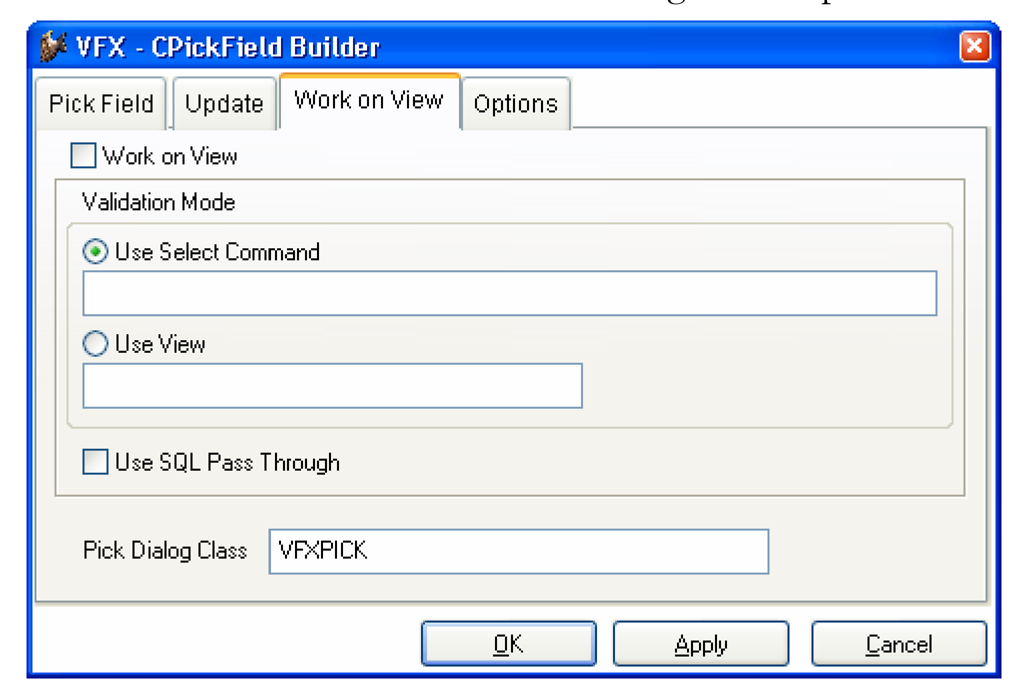
Update Target Fields. Weisen sie die Zielfelder zu. Wenn Sie mehrere Werte eingeben, so müssen diese durch Semikolon getrennt werden.

OK. Die eingestellten Optionen werden in das ausgewählte Auswahllisten-Objekt eingefügt.

Apply. Macht das gleiche wie *OK,* jedoch wird der VFX-Pickfield-Builder nicht beendet.

Cancel. Bricht die Arbeit mit dem VFX – CPickField Builder ab. Alle Eingaben werden verworfen.

Auf der Seite *Work on View* stehen die folgenden Optionen zur Verfügung:



Work on View. Wenn die Daten, aus denen Sie auswählen aus einer Ansicht stammen, markieren Sie dieses Kontrollkästchen.

Use Select Command: Wahlweise kann ein Select-Befehl oder eine Ansicht zur Überprüfung der Benutzzereingabe verwendet werden. Wenn Sie einen Select-Befehl verwenden, muss durch eine Where-Klausel sichergestellt sein, dass maximal ein Wert zurückgegeben wird. Beispiel: *„select customer_id from lv_customer where customer_id = trim(this.txtField.Value)"*

Use View: Wahlweise kann ein Select-Befehl oder eine Ansicht zur Überprüfung der Benutzzereingabe verwendet werden. Wenn Sie eine Ansicht verwenden, geben Sie hier den Namen der Ansicht ein. Die Where-Klausel der Ansicht muss sicherstellen, dass maximal ein Wert zurückgegeben wird.

Use SQL Pass Through: Wenn Sie dieses Kontrollkästchen markieren, wird der in der Ansicht enthaltene Select-Befehl von VFX ausgelesen und per SQL Pass Through an die Remote-Datenquelle gesendet.

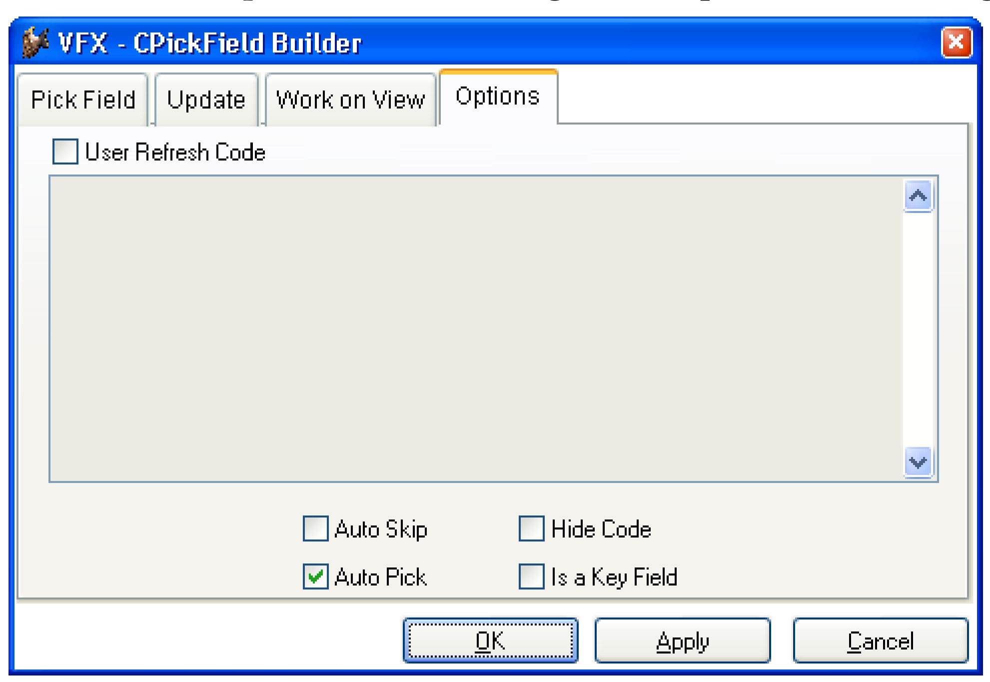
Pick Dialog Class: Hier kann eine eigene Klasse für das Auswahllisten-Steuerelement verwendet werden. Beachten Sie, dass die Klasse von der Klasse CPickField abgeleitet sein muss.

OK. Die eingestellten Optionen werden in das ausgewählte Auswahllisten-Objekt eingefügt.

Apply. Macht das gleiche wie *OK*, jedoch wird der VFX – CPickField Builder nicht beendet.

Cancel. Bricht die Arbeit mit dem VFX – CPickField Builder ab. Alle Eingaben werden verworfen.

Auf der Seite *Options* stehen die folgenden Optionen zur Verfügung:



User Refresh Code. Manchmal benötigen Sie speziellen Code in der *Refresh()*-Methode des Auswahllisten-Containers.

Auto Skip. Markieren Sie diese Option, wenn Sie automatisch zum nächsten Steuerelement springen wollen, nachdem Sie einen Wert aus der Auswahlliste ausgewählt haben. Dadurch wird die *CPickField*-Eigenschaft *lUseTab* auf *.T.* gesetzt.

Auto Pick. Markieren Sie diese Option, wenn Sie automatisch die Auswahlliste aufrufen wollen, wenn der Benutzer einen falschen Wert eingegeben hat. Dadurch wird die *CPickField*-Eigenschaft *lAutoPick* auf *.T.* gesetzt.

Hide Code. Markieren Sie diese Option, wenn Sie das Eingabefeld in der Auswahlliste verstecken wollen. Dadurch wird die *CPickField*-Eigenschaft *lHideCode* auf *.T.* gesetzt. Der Benutzer kann keinen Wert eingeben, sondern nur aus der Auswahlliste auswählen.

Is a Key Field. Markieren Sie diese Option, wenn Sie dieses Auswahllistenfeld als Schlüsselfeld definieren wollen. Ein Schlüsselfeld ist nur zugänglich während Sie einen neuen Datensatz anlegen (so wie die Textfeld-Klasse *ckeyfield*). Dadurch wird die *CPickField*-Eigenschaft *lKeyField* auf *.T.* gesetzt.

OK. Die eingestellten Optionen werden in das ausgewählte Auswahllisten-Objekt eingefügt.

Apply. Macht das gleiche wie *OK,* jedoch wird der VFX – CPickField Builder nicht beendet.

Cancel. Bricht die Arbeit mit dem VFX – CPickField Builder ab. Alle Eingaben werden verworfen.

8.6.4. Test und Verfeinerung des Formulars

Starten Sie Ihre Anwendung, wählen Sie im Öffnen-Dialog Ihr neu erstelltes Formular und starten Sie es mit einem Mausklick. Testen Sie es und prüfen Sie, wo Ihr Formular erweitert werden muss.

8.6.5. Nächste Schritte

Um mit dem VFX-Formular-Builder besser vertraut zu werden, lohnt es sich, einige Formulare zu generieren. Beginnend mit einfachen Formularen, später auch Formularen, die andere Formulare aufrufen.

Nachdem Sie mit dem Erstellen von Standard-VFX-Datenbearbeitungs-Formularen vertraut sind, können Sie sich den *1:n-Datenbearbeitungs-Formularen* zuwenden.

8.7. 1:n Formulare

Das 1:n-Formular ist eine Weiterentwicklung des Standard-VFX-Datenbearbeitungs-Formulars. Das bedeutet, dass Sie auf einem einzigen Formular die normalen Datenbearbeitungsfunktionen haben können und ein Grid mit den Child-Datensätzen zu dem aktuell angezeigten Hauptdatensatz haben. VFX erlaubt es Ihnen, auch mehrere Child-Tabellen zu einer Haupttabelle auf mehreren Seiten eines Seitenrahmens zu bearbeiten. Wenn Sie viele Eingabefelder in Ihrer Child-Tabelle haben, können Sie die Felder auf mehrere Seiten eines Seitenrahmens verteilen. Das erlaubt Ihnen, eine große Anzahl verschiedenster Anwendungen abzudecken ohne wirklich programmieren zu müssen. Alles was Sie wissen müssen ist, wie man ein 1:n-Formular erstellt, die zugehörige Datenbank einrichtet und durch welche Felder die Haupttabelle und die Child-Tabelle miteinander verbunden sind. Lassen Sie uns ein einfaches Beispiel betrachten:

8.7.1. Ergebnis

Bitte lesen Sie im Kapitel *Diskussion des VFX-1:n-Datenbearbeitungs-Formulars* weiter unten in diesem Handbuch nach, um eine Vorstellung über die Bedienung von 1:n-Formularen zu bekommen, die mit VFX erstellt wurden.

8.7.2. Erstellen eines neuen Formulars

Starten Sie aus dem VFX-Menü den VFX – Form Wizard und erstellen Sie ein Formular basierend auf der Klasse *cOneToMany*.

8.7.3. Einrichten der Datenumgebung

Wie schon weiter oben in diesem Handbuch beschrieben, müssen Sie die Datenbank Ihrer Anwendung einrichten. Definieren Sie Ihre Tabellen, Felder und Indexschlüssel sowie die Feldüberschriften. Die VFX-Builder benutzen diese Informationen, sodass Sie die Überschriften nicht nochmals eingeben müssen.

Bevor Sie ein 1:n-Formular erstellen, sollten Sie die Grundlagen des Datenbank-Designs und insbesondere 1:n-Beziehungen beherrschen. In 1:n-Beziehungen stellen Sie die Verbindung von einem Hauptdatensatz zu den Child-Datensätzen her. Ein gutes Beispiel für eine 1:n-Beziehung ist die Verbindung zwischen Aufträgen (Haupttabelle) und Auftragspositionen (Child-Tabelle) in jedem Auftragsbearbeitungssystem.

ANMERKUNG: Wenn Sie die referentielle Integrität (RI) nicht manuell mit Hilfe der VFX-Methoden wie *OnPostDelete()* herstellen wollen, ist es sinnvoll, den RI-Code im Datenbank-Designer anzulegen, bevor Sie mit der Erstellung von 1:n-Formularen beginnen. Wenn Sie diese Arbeit manuell erledigen wollen, müssen Sie den Code für das Löschen von Hauptdatensätzen und den zugehörigen Child-Datensätzen von Hand schreiben. Wenn Sie außerdem die Änderung des Schlüsselfeldes in der Haupttabelle erlauben, müssen Sie auch den Code schreiben, um die Child-Datensätze zu aktualisieren.

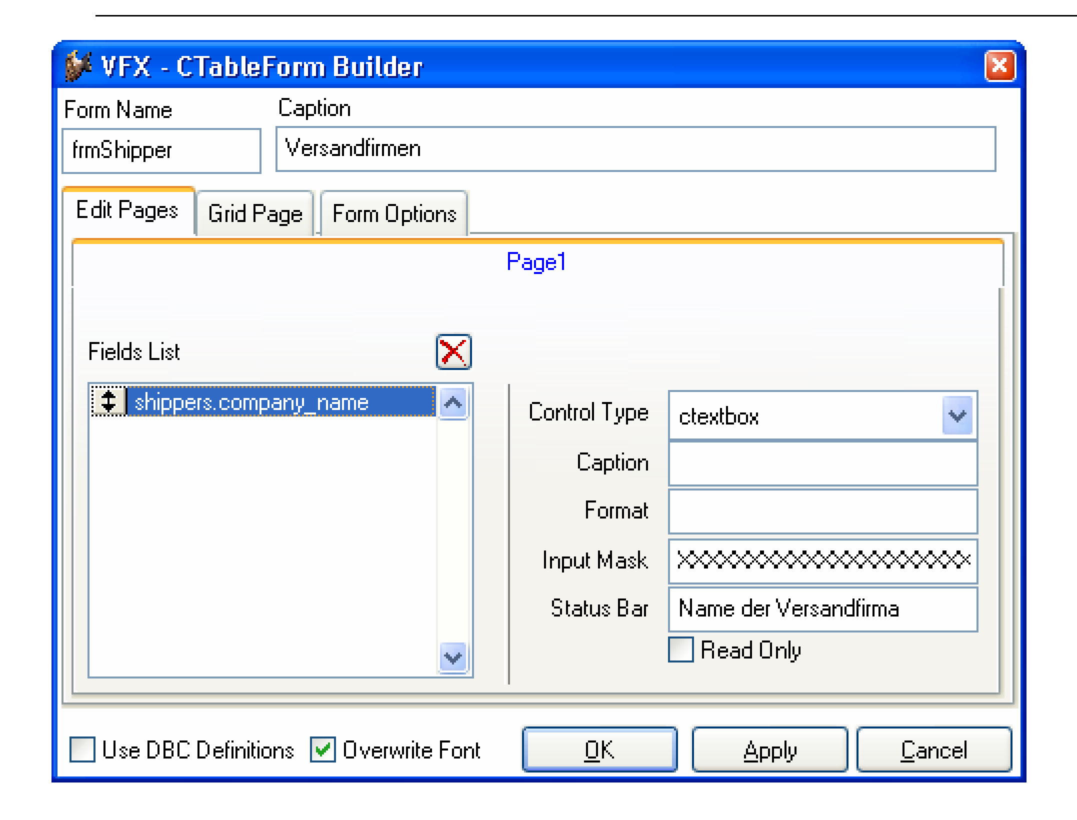
Richten Sie die Datenumgebung des Formulars ein, das Sie erstellen wollen. Der VFX – COneToMany Builder verwendet diese Informationen automatisch beim Erstellen des 1:n-Formulars.

Der VFX – COneToMany Builder hilft Ihnen bei der Erstellung von anspruchsvollen 1:n-Formularen, fast ohne zu programmieren. Wenn Sie die 1:n-Beziehung zwischen der Haupttabelle und der Child-Tabelle hergestellt haben, können Sie 1:n-Formulare genauso einfach erstellen wie Standard-VFX-Datenbearbeitungsformulare. Wenn Sie mehrere Child-Tabellen mit ei-

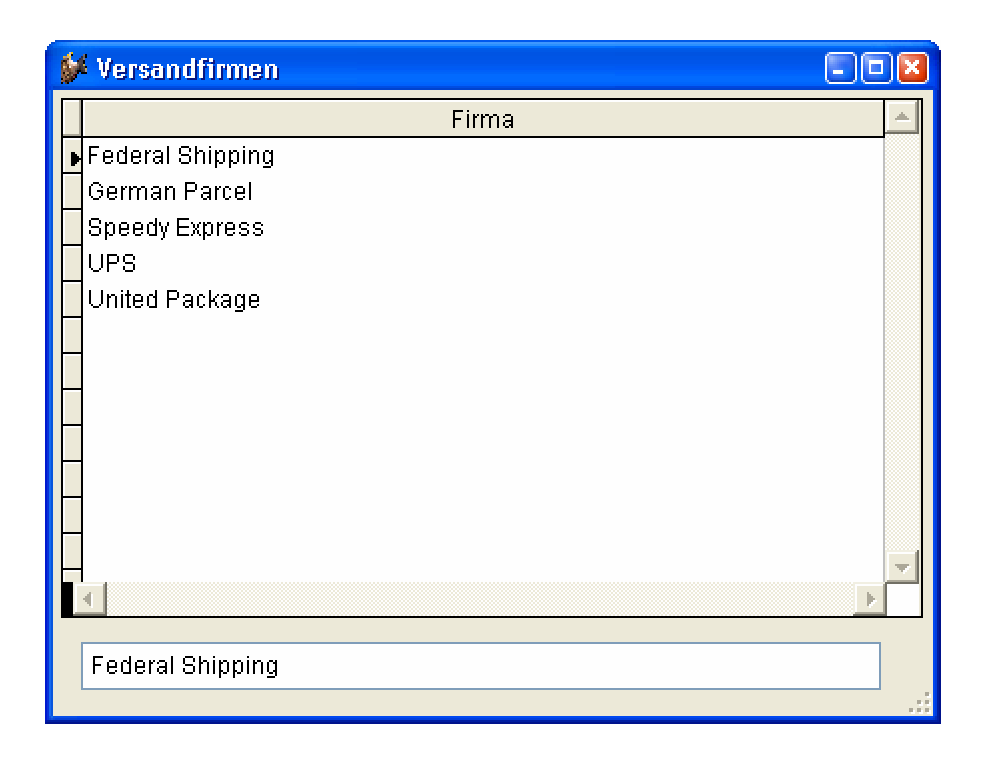
ner Haupttabelle verbinden wollen, müssen Sie von jeder Child-Tabelle eine Beziehung zu der Haupttabelle herstellen.

WICHTIG: Denken Sie daran, den *InitialSelectedAlias* in der Datenumgebung anzugeben sowie die *Order*-Eigenschaft in den Tabellen Ihrer Datenumgebung. Außerdem müssen Sie die 1:n-Beziehung zwischen der Haupttabelle und der Child-Tabelle herstellen. Ansonsten wird Ihr Formular nicht so funktionieren, wie Sie es erwarten!

8.8. Der VFX – CTableForm Builder



Eine weitere Formularart ist die CTableForm. Bei diesem Formular werden das Listen-Grid und die Steuerelemente nebeneinander oder untereinander dargestellt. Es eignet sich daher insbesondere für Formulare mit nur wenigen Eingabefeldern.



8.9. Der VFX - COneToMany Builder

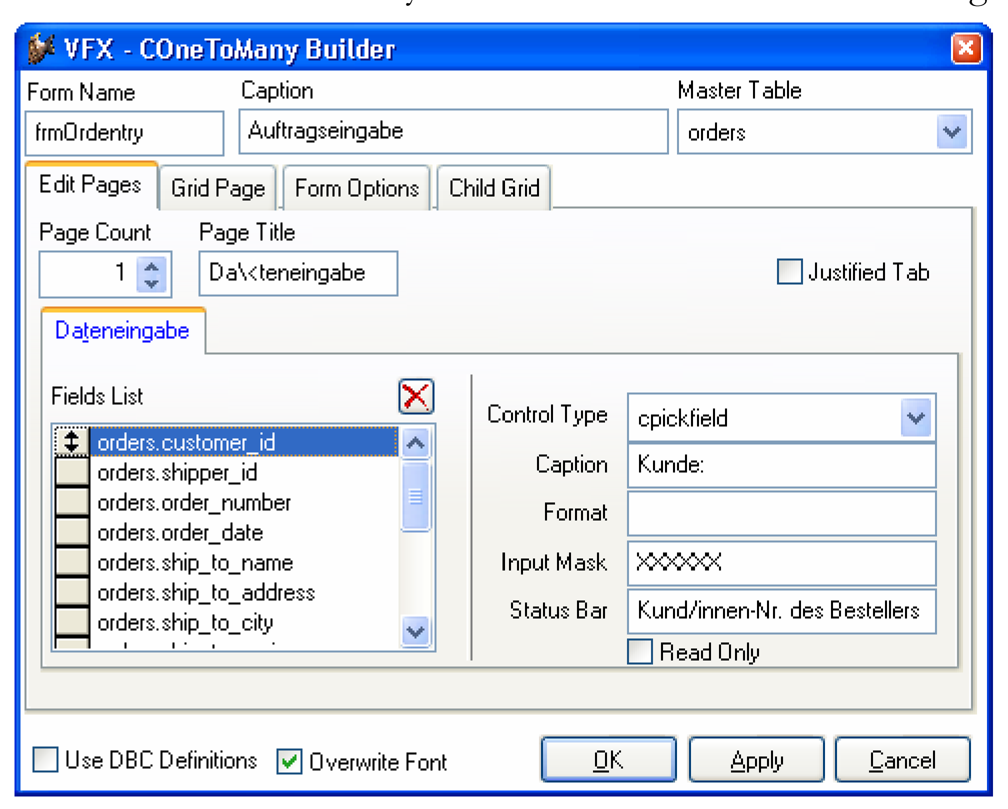
8.9.1. Aufruf des VFX – COneToMany Builder

Um den VFX-1:n-Formular-Builder aufzurufen, bewegen Sie die Maus auf den weißen Hintergrund des Formular-Designers, drücken Sie die rechte Maustaste und wählen Sie Builder.

ANMERKUNG: Wenn Sie eine Meldung erhalten, dass es keinen Builder für das ausgewählte Objekt gibt oder wenn ein Standard-Visual FoxPro-Builder erscheint, prüfen Sie, ob Sie das Formularobjekt ausgewählt haben. Ein verbreiteter Fehler ist, den Seitenrahmen anstelle des Formularobjektes auszuwählen. Prüfen Sie das angezeigte Objekt im Eigenschaftsfenster, wenn Sie nicht sicher sind, ob das Formularobjekt ausgewählt ist.

8.9.2. Die Bedienung des VFX - COneToMany Builder

Der VFX - COneToMany Builder hat eine intuitive Bedienung.



Bearbeiten Sie zunächst die folgenden Optionen:

Form Name. Siehe Beschreibung im Kapitel *Der VFX-Formular-Builder.*

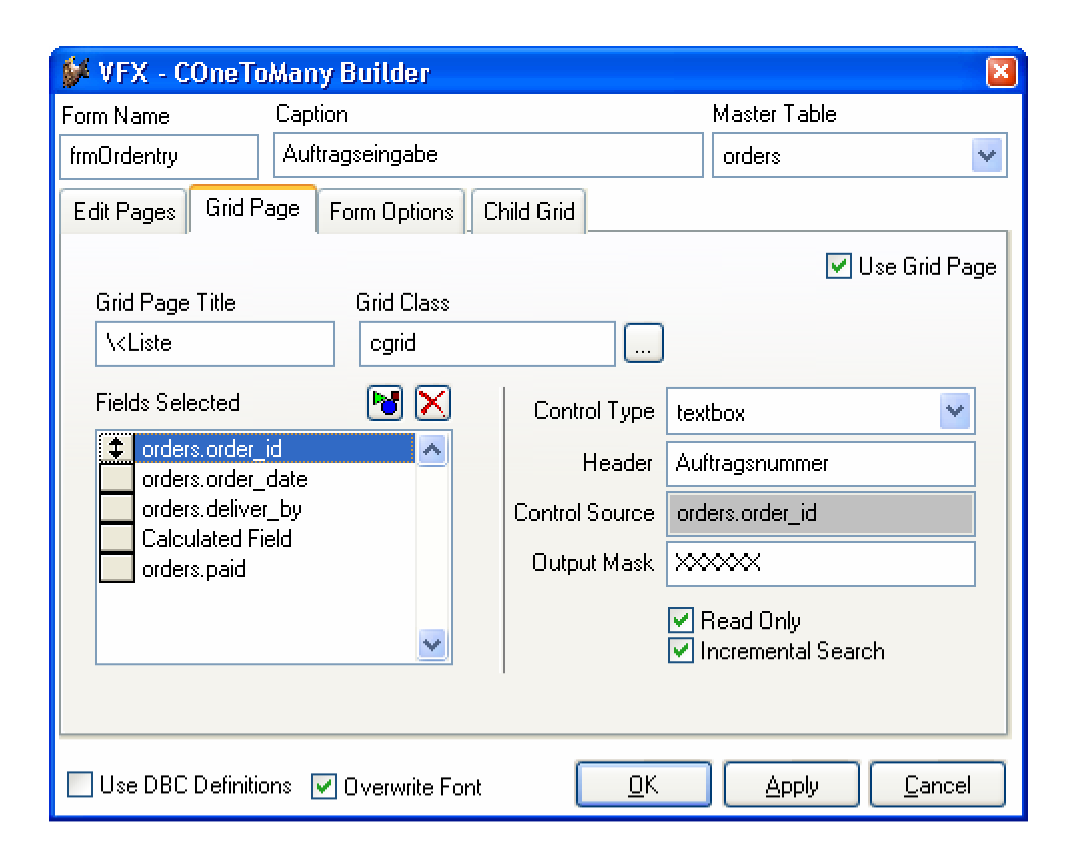
Caption. Siehe Beschreibung im Kapitel *Der VFX-Formular-Builder.*

Master Table. Name der Haupttabelle oder Ansicht.

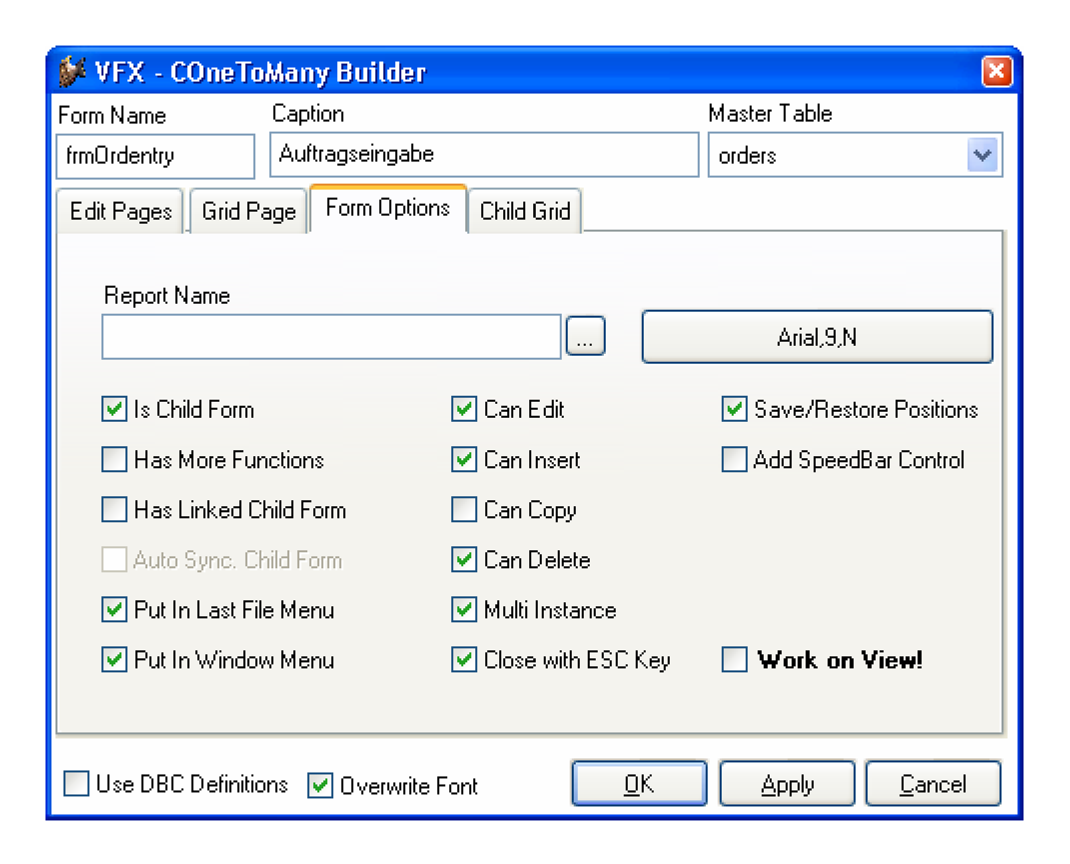
Als nächstes bearbeiten Sie den Seitenrahmen mit den Seiten *Edit Pages*, *Grid Page, Form Options* und *Child Grid*:

Auf der Seite mit dem Namen *Edit Pages* sehen Sie die gleichen Bedienungselemente wie im VFX-Formular-Builder, der weiter oben in diesem Handbuch beschrieben wurde. Hier legen Sie die Eigenschaften der Bearbeitungsseiten für die Haupttabelle fest:

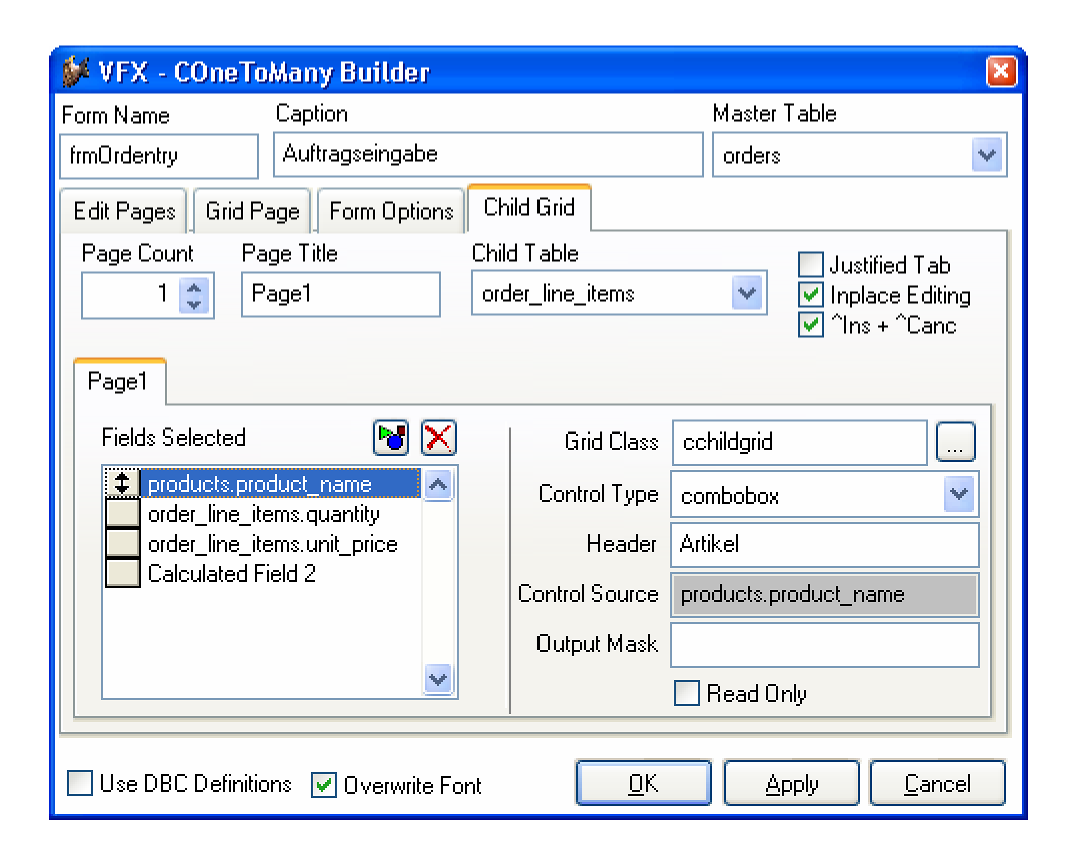
Auf der Seite mit dem Namen *Grid Page* sehen Sie die gleichen Bedienungselemente wie im VFX-Formular-Builder, der weiter oben in diesem Handbuch beschrieben wurde. Hier beschreiben Sie die Eigenschaften des Grids für die Haupttabelle:



Auf der Seite mit dem Namen *Form Options* sehen Sie die gleichen Bedienungselemente wie im VFX-Formular-Builder, der weiter oben in diesem Handbuch beschrieben wurde. Hier wählen Sie die Optionen für das 1:n-Formular:



Auf der Seite mit dem Namen *Child Grid* geben Sie an, wie das oder die Grids mit den Child-Daten aussehen sollen:



Page Count. Geben Sie ein, wie viel Child-Grids Ihr Formular haben soll. Für die meisten 1:n-Formulare wird ein Grid ausreichen. Wenn Sie mehrere Child-Tabellen haben, werden Sie diese über mehrere Seiten verteilen wollen. Entsprechend der Anzahl der Seiten, die Sie gewählt haben, erscheint der Seitenrahmen des Formular-Builders mit der gewählten Anzahl von Seiten. Wenn Sie zwei Seiten einstellen, hat der Seitenrahmen zwei Seiten, wenn Sie drei Seiten einstellen, hat der Seitenrahmen drei Seiten usw.

Page Title. Geben Sie die Überschrift für das aktuell gewählte Child-Grid an. Wenn Sie die Überschrift für die zweite Seite eingeben wollen, drücken Sie auf die zweite Seite. Der VFX – COneToMany Builder zeigt sofort den eingegebenen Text als Überschrift der jeweiligen Seite an.

Child Table. Geben Sie die Datenquelle für Ihr Child-Grid an. Achtung: Es ist sehr wichtig, diese Einstellung zu machen. Wenn Sie diese Eigenschaft nicht einstellen, wird Ihr Formular nicht richtig funktionieren.

Justified Tab. Markieren Sie dieses Kontrollkästchen, wenn die Seitenüberschriften justiert sein sollen. Ansonsten haben die Überschriften eine variable Länge und füllen nicht die Breite des Seitenrahmens.

Inplace Editing. Markieren Sie diese Option, wenn Sie Daten in das Child-Grid eingeben wollen, was normalerweise der Fall ist.

^Ins+^Canc. Markieren Sie diese Option, wenn Sie die Möglichkeit haben wollen, mit Strg+Einfg Datensätze einzufügen und mit Strg+Entf Datensätze im Child-Grid zu löschen.

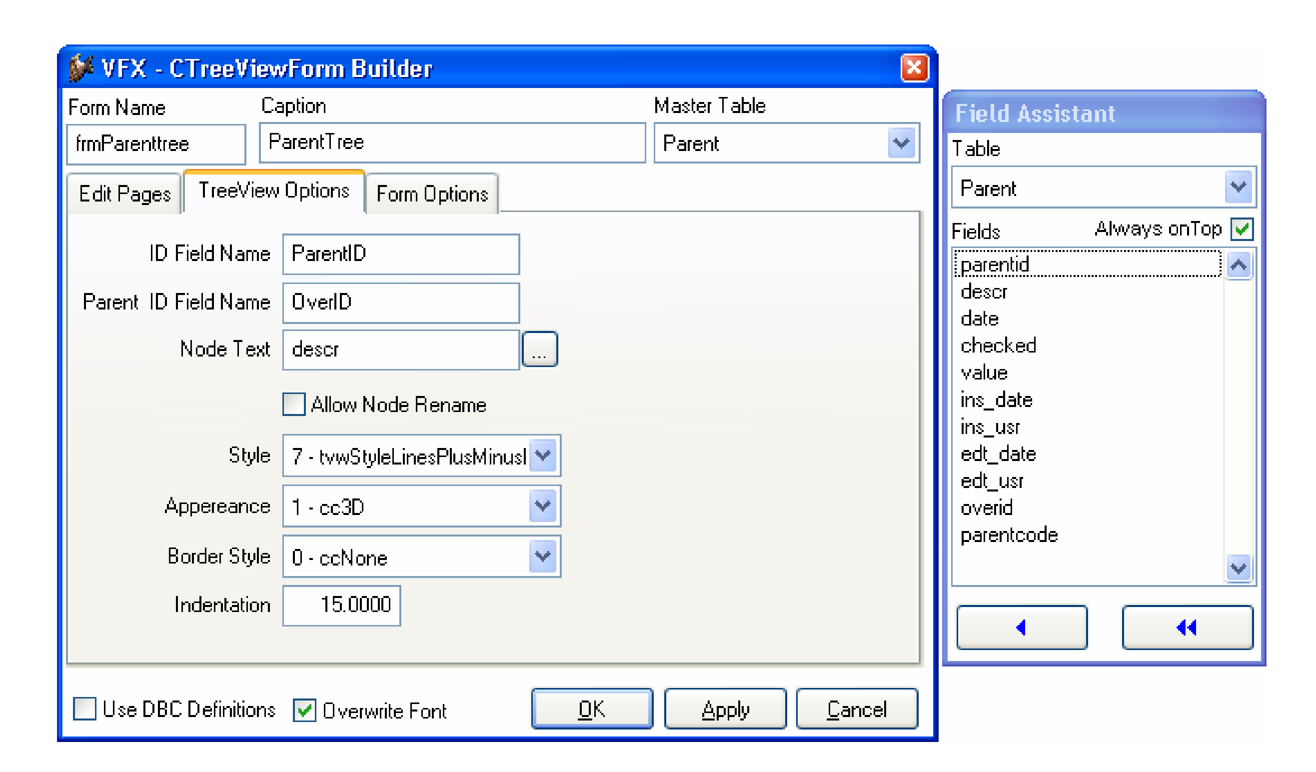
Die anderen Optionen sind mit denen auf der Grid-Seite des VFX-Formular-Builders identisch.

8.10. Die Klasse cTreeViewForm

Der Haupteinsatzzweck dieser Klasse ist die Darstellung von Daten aus einer Tabelle in einer Baumstruktur. Die Baumstruktur gibt dem Endanwender einen kompletten Überblick über die hierarchischen Beziehungen in einer Tabelle.

Diese Klasse basiert auf der Klasse cDataFormPage (Vfxform.vcx) und enthält ein Treeview-Steuerelement aus der Klasse cTreeView (Vfxappl.vcx). Die Klasse kombiniert die Funktionalität von cDataFormpage mit den Möglichkeiten der hierarchischen Datenpräsentation in einer Baumstruktur. Wenn ein Eintrag im Treeview-Steuerelement ausgewählt wird, wird der Datensatzzeiger in der zugrunde liegenden Tabelle mitgeführt und der Anwender kann die Daten im rechten Teil des Formulars bearbeiten.

Mit dem VFX – CTreeViewForm Builder können sehr schnell Formulare basierend auf der Klasse cTreeViewForm erstellt und alle benötigten Eigenschaften können eingestellt werden.



Der Builder arbeitet so ähnlich wie der VFX – CDataFormPage Builder. Die Einstellungen können auf den Seiten Edit Pages und Form Options genauso gemacht werden, wie im VFX – CDataFormPage Builder. Zusätzlich müssen die Einstellungen für das Treeview-Steuerelement auf der Seite TreeViewOptions gemacht werden. Es müssen zwei Arten von Einstellungen für das Treeview-Steuerelement gemacht werden.

8.10.1. Einstellungen zur Datenanbindung des TreeView-Steuerelements

IDFieldName – Hier wird der Name des Feldes mit dem Primärschlüssel der Bearbeitungstabelle eingetragen.

ParentIDFieldName – Diese Eigenschaft enthält den Namen des Feldes, in dem der Primärschlüssel des Parent-Datensatzes gespeichert ist.

NodeText – Hier kann entweder der Name eines Feldes, das einen Beschreibungstext enthält eingetragen werden oder es wird ein Ausdruck eingetragen, der zur Laufzeit evaluiert wird und dessen Rückgabewert als Bezeichnung in der Baumstruktur angezeigt wird. Wenn ein Feldname verwendet wird, kann dem Anwender erlaubt werden die Bezeichnung direkt im Treeview-Steuerelement zu ändern. Dies hängt vom Wert der Eigenschaft *AllowNodeRename* ab. Wenn *AllowNodeRename* auf .T. gesetzt ist, kann der Anwender die Bezeichnungen im Treeview-Steuerelement ändern. Dabei werden die Daten im zugrunde liegenden Tabellenfeld automatisch aktualisiert.

AllowNodeRename – Über diese Eigenschaft wird gesteuert, ob der Anwender die Bezeichnung im Treeview-Steuerelement ändern kann. Die Bearbeitung der Bezeichnung im Treeview-Steuerelement ist nur möglich, wenn die Bezeichnung auf einem einzelnen Tabellenfeld basiert. Dieses Tabellenfeld wird bei der Bearbeitung automatisch aktualisiert.

8.10.2. Layout-Einstellungen des TreeView-Steuerelements

Diese Einstellungen entsprechen denen des TreeView-ActiveX-Steuerelements.

Style: 0 - tvwStyleText

1 - tvwStylePictureText

2 - tvwStylePlusMinusText

3 - tvwStylePlusMinusPictureText

4 - tvwStyleLinesText

5 - tvwStyleLinesPictureText

6 - tvwStyleLinesPlusMinusText

7 - tvwStyleLinesPlusMinusPictureText

Appearance: 0 - ccFlat

1 - cc3D

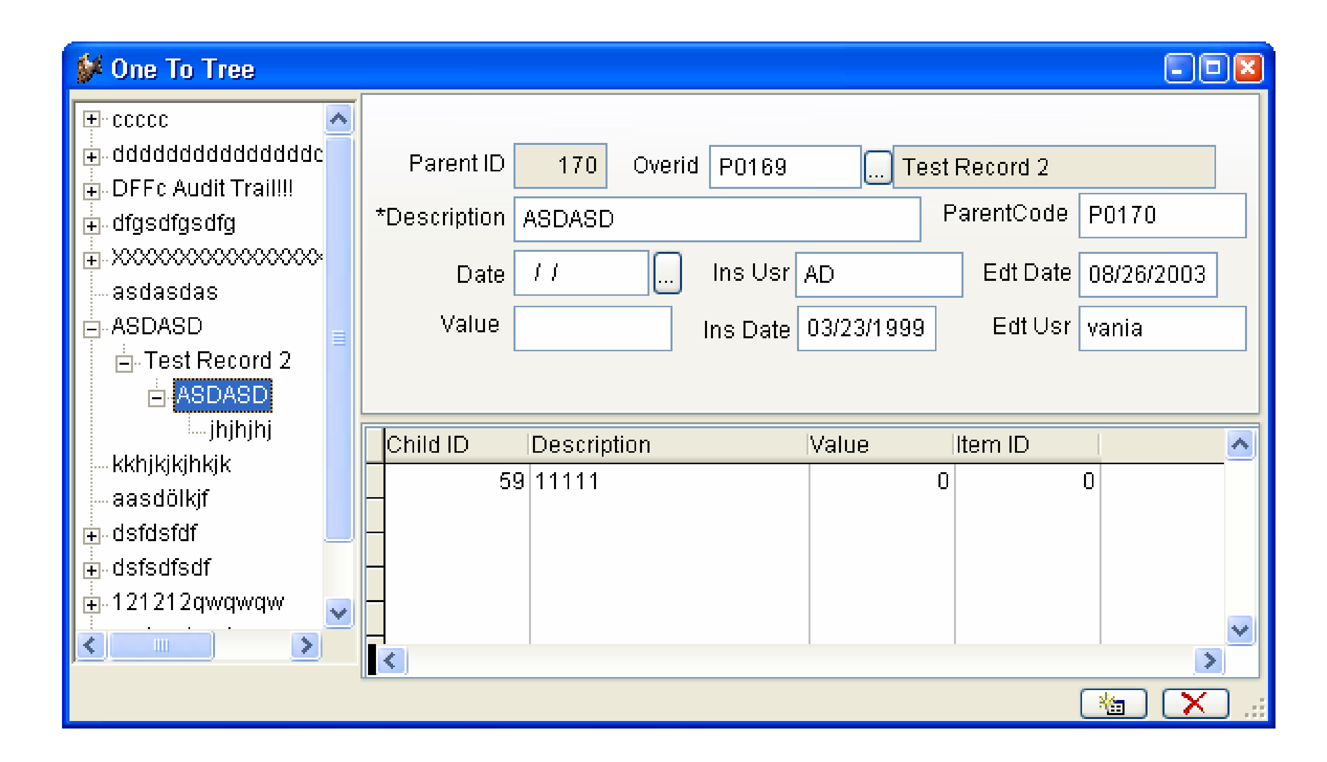
BorderStyle: 0 - ccNone

1 - ccFixedSingle

Indentation: Diese Eigenschaft bestimmt die Breite des Einzugs der Knoten.

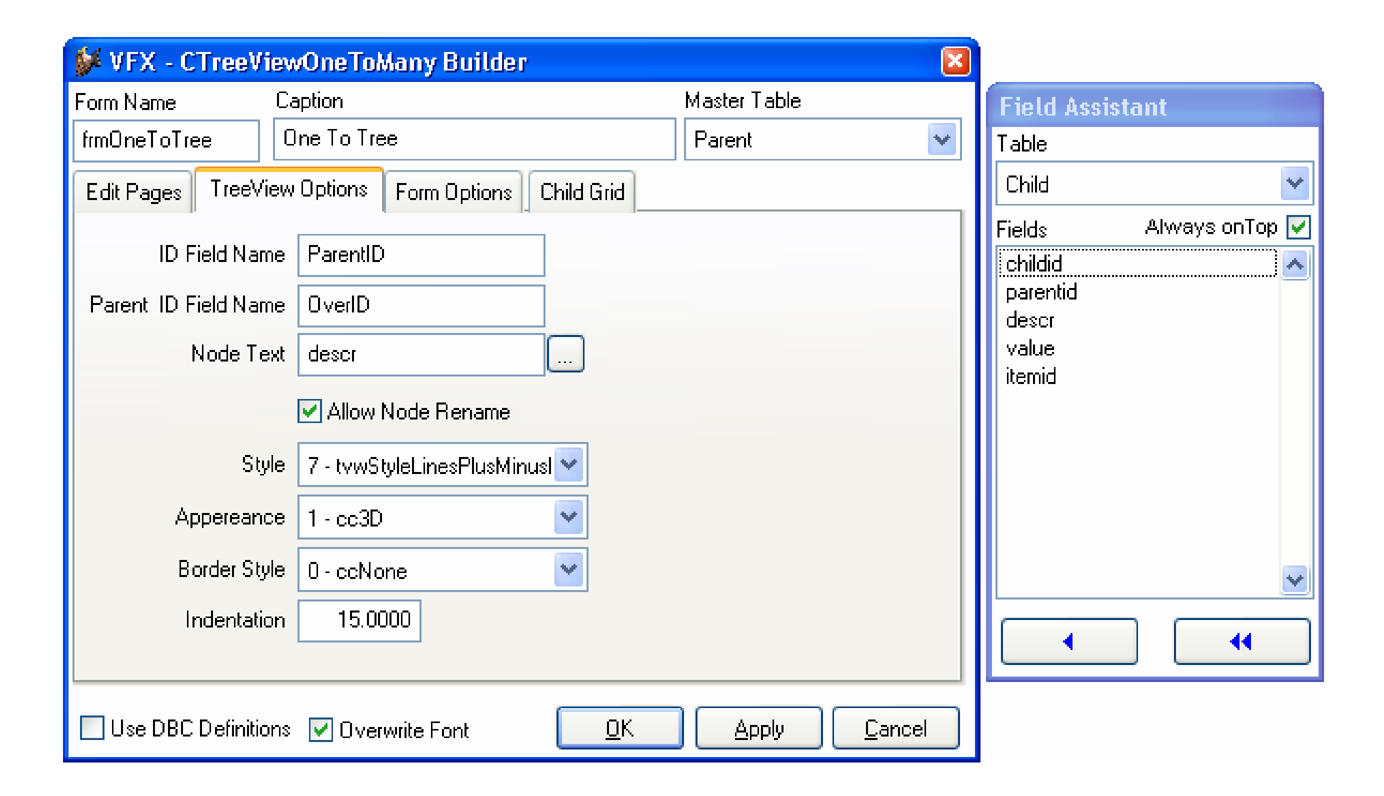
8.11. Die Klasse cTreeViewOneToMany

Der Haupteinsatzzweck dieser Klasse ist die Darstellung der Daten aus einer Tabelle in einer Baumstruktur zusammen mit der leistungsfähigen Funktionalität, die die cOneToMany-Klasse dem Entwickler bietet. Die Baumstruktur gibt dem Anwender den kompletten Überblick über die hierarchischen Datenbeziehungen.



Diese Klasse basiert auf der Klasse cOneToMany (Vfxform.vcx) und enthält ein Treeview-Steuerelement aus der Klasse cTreeView (Vfxappl.vcx). Die Klasse kombiniert die Funktionalität von cOneToMany mit den Möglichkeiten der hierarchischen Datenpräsentation in einer Baumstruktur. Wenn ein Eintrag im Treeview-Steuerelement ausgewählt wird, wird der Datensatzzeiger in der zugrunde liegenden Tabelle mitgeführt und der Anwender kann die Daten im rechten Teil des Formulars bearbeiten. Zusätzlich können die Child-Daten im unteren Teil des Formulars bearbeitet werden.

Mit dem VFX – CTreeViewOneToMany Builder können sehr schnell Formulare basierend auf der Klasse cTreeViewOneToMany erstellt und alle benötigten Eigenschaften eingestellt werden.



Dieser Builder arbeitet so ähnlich wie der VFX – COneToMany Builder. Die Einstellungen auf den Seiten Edit Pages, Form Options und Child Grid werden genauso gemacht, wie bei Formularen basierend auf der Klasse cOneToMany. Zusätzlich müssen die Einstellungen für das Treeview-Steuerelement auf der Seite TreeViewOptions gemacht werden.

Die Einstellungen erfolgen genauso wie beim VFX –CTreeViewForm Builder.

8.11.1. Einstellungen zur Datenanbindung des TreeView-Steuerelements

IDFieldName – Hier wird der Name des Feldes mit dem Primärschlüssel der Bearbeitungstabelle eingetragen.

ParentIDFieldName – Diese Eigenschaft enthält den Namen des Feldes, in dem der Primärschlüssel des Parent-Datensatzes gespeichert ist.

NodeText – Hier kann entweder der Name eines Feldes, das einen Beschreibungstext enthält eintragen werden oder es wird ein Ausdruck eingetragen, der zur Laufzeit evaluiert wird und dessen Rückgabewert als Bezeichnung in der Baumstruktur angezeigt wird. Wenn ein Feldname verwendet wird, kann dem Anwender erlaubt werden die Bezeichnung direkt im Treeview-Steuerelement zu ändern. Dies hängt vom Wert der Eigenschaft *AllowNodeRename* ab. Wenn *AllowNodeRename* auf .T. gesetzt ist, kann der Anwender die Bezeichnungen im Treeview-

Steuerelement ändern. Dabei werden die Daten im zugrunde liegenden Tabellenfeld automatisch aktualisiert.

AllowNodeRename – Über diese Eigenschaft wird gesteuert, ob der Anwender die Bezeichnung im Treeview-Steuerelement ändern kann. Die Bearbeitung der Bezeichnung im Treeview-Steuerelement ist nur möglich, wenn die Bezeichnung auf einem einzelnen Tabellenfeld basiert. Dieses Tabellenfeld wird bei der Bearbeitung automatisch aktualisiert.

8.11.2. Layout-Einstellungen des TreeView-Steuerelements

Diese Einstellungen entsprechen denen des TreeView-ActiveX-Steuerelements.

Style: 0 - tvwStyleText

1 - tvwStylePictureText

2 - tvwStylePlusMinusText

3 - tvwStylePlusMinusPictureText

4 - tvwStyleLinesText

5 - tvwStyleLinesPictureText

6 - tvwStyleLinesPlusMinusText

7 - tvwStyleLinesPlusMinusPictureText

Appearance: 0 - ccFlat

1 - cc3D

BorderStyle: 0 - ccNone

1 - ccFixedSingle

Indentation: Diese Eigenschaft bestimmt die Breite des Einzugs der Knoten.

8.12. Der VFX – CChildGrid Builder

Der VFX – CChildGrid Builder erlaubt Ihnen, die Funktionalität der Child-Grids zu erweitern. Benutzen Sie diesen Builder, um die Felder für das Grid zusammenzustellen oder um den Code der Methode *OnPostInsert()* zu bearbeiten. Diese Methode wird immer dann ausgeführt, wenn dem Child-Grid ein neuer Datensatz hinzugefügt wurde. Ähnlich wie im Standard-VFX-

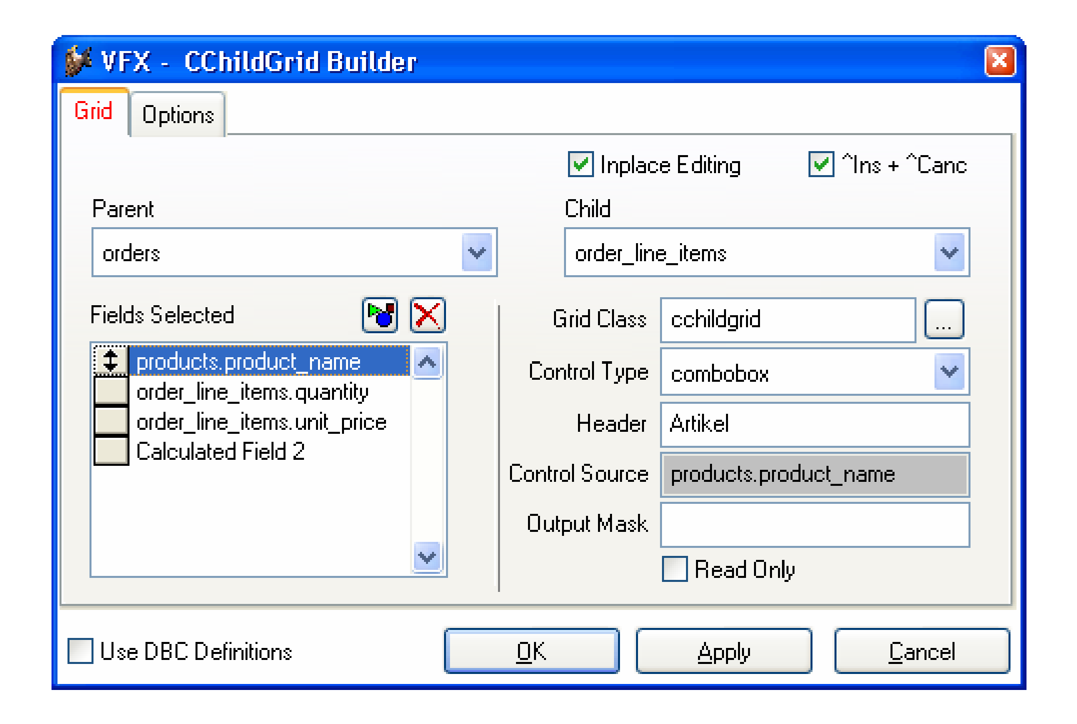
Datenbearbeitungsformular stehen Ihnen hier die folgenden Ereignisse zur Verfügung:

- OnPreInsert()
- OnInsert()
- OnPostInsert()

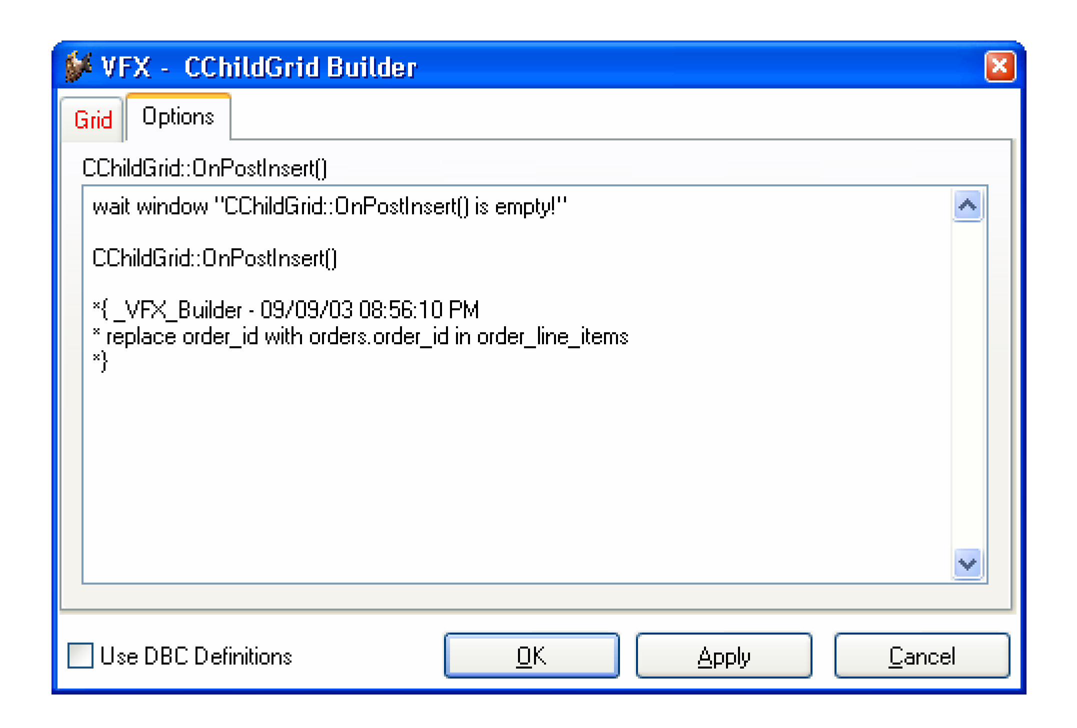
In der *OnPostInsert()*-Methode des Child-Grids müssen Sie das Feld der Child-Tabelle ausfüllen, das die Verknüpfung zur Haupttabelle herstellt. Normalerweise benötigen Sie dafür folgenden Code:

```
REPLACE <ChildLinkField> WITH <Master.MasterField> IN <ChildTable>
```

Der VFX – CChildGrid Builder ist wie folgt zu bedienen. Auf der ersten Seite mit dem Namen *Grid* können Sie das Child-Grid, wie weiter oben in diesem Abschnitt beschrieben, anpassen:



Auf der zweiten Seite mit dem Namen *Options* können Sie den Code der *OnPostInsert()*-Methode bearbeiten, um das Feld der Child-Tabelle mit dem Wert der Haupttabelle zu füllen.



ANMERKUNG: Der Grund, aus dem der VFX-Builder den Code der *OnPostInsert()*-Methode nicht automatisch generieren kann, ist, dass Sie zusammengesetzte Schlüssel verwenden könnten oder mehreren Feldern in der Child-Tabelle Werte zuweisen möchten. Wenn einfache Schlüssel verwendet werden, ist der generierte Code in der Regel richtig und Sie brauchen nur das Kommentarzeichen am Zeilenanfang zu entfernen.

8.13. Die Klasse cPickAlternate

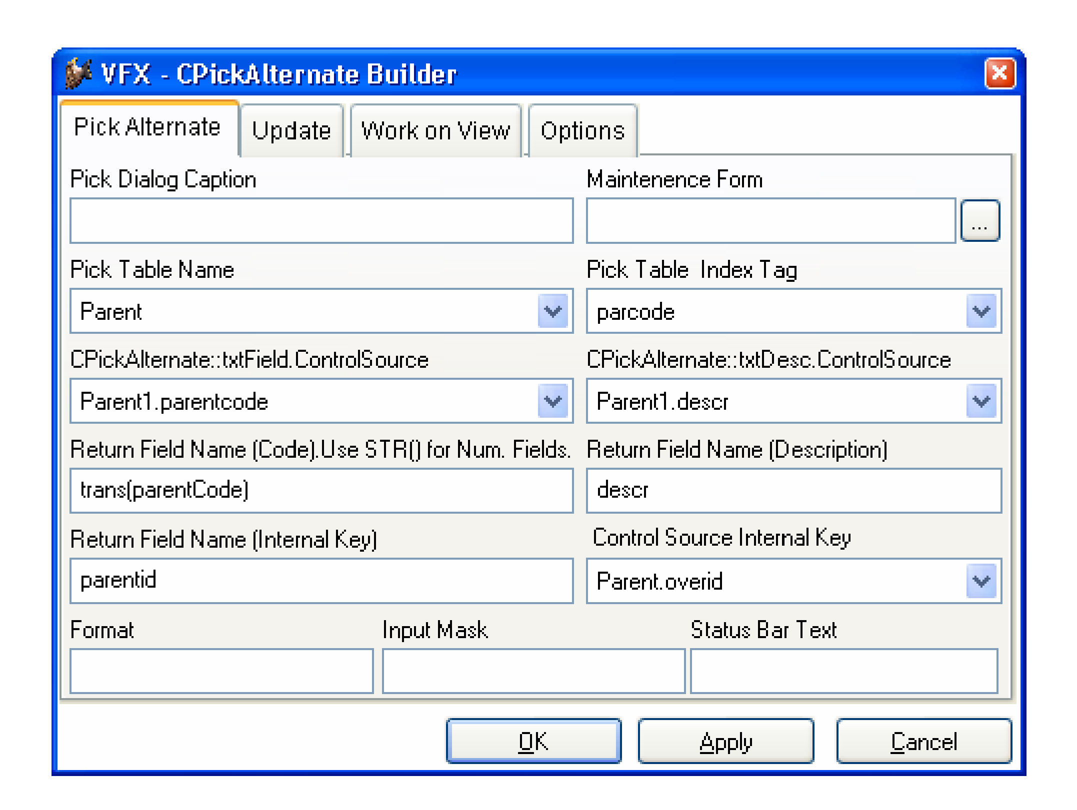
Ähnlich zum cPickField-Steuerelement kann die Klasse cPickAlternate verwendet werden um eine Benutzereingabe zu verifizieren. Es kann eine Auswahlliste aufgerufen werden, die dem Anwender erlaubt einen Wert aus einer Liste auszuwählen. Bei Verwendung der Klasse cPickAlternate wird der Primärschlüssel des ausgewählten Datensatzes in der Bearbeitungstabelle gespeichert während der Benutzer einen Wert aus einem anderen Feld aus der Auswahltabelle angezeigt bekommt.

Das cPickAlternate-Steuerelement ist einer Combobox zu bevorzugen, wenn aus einer Tabelle mit vielen Datensätzen ausgewählt werden soll. Der Einsatz ist auch sinnvoll, wenn der vom Anwender eingegebene Wert nicht dem Schlüssel der Auswahltabelle entspricht. Das Ziel dieser Klasse ist es dem Anwender eine einfach zu bedienende Schnittstelle zu geben, die es erlaubt ihm bekannte Werte einzugeben anstelle von vom Programm generierten Primärschlüsseln. Der vom Anwender eingegebene Wert wird verwendet um den dazugehörigen Datensatz in der Auswahltabelle zu finden. Wenn der ge-

suchte Datensatz gefunden ist, wird als Rückgabewert der Primärschlüssel an das cPickAlternate-Steuerelement zurückgegeben.

Diese Klasse basiert auf der Klasse cPickField und erbt alle ihre Eigenschaften und Methoden. Zusätzlich hat diese Klasse die neue Eigenschaft cControlSourceInternalKey in die der Name des Feldes der Bearbeitungstabelle mit dem Fremdschlüssel eingetragen wird. Dieser Fremdschlüssel entspricht dem Primärschlüssel aus der Auswahltabelle.

Mithilfe des VFX – CPickAlternate Builder können die Eigenschaften dieser Klasse einfach eingestellt werden.



Pick Table Name – Hier kann der Name der Auswahltabelle aus einer der Datenquellen der Datenumgebung ausgewählt werden.

Pick Table Index Tag – Dies ist der Name des Indexschlüssels, der verwendet wird um in der Auswahltabelle zu suchen. Dieser Indexschlüssel entspricht dem Wert des Eingabefeldes.

CPickAlternate.txtField.ControlSource – Die Controlsource des Eingabefeldes. Dieses Feld muss aus der Auswahltabelle stammen.

CpickAlternate.txtDesc.ControlSource – Der Name des Beschreibungsfeldes. Der Wert wird nach der erfolgreichen Überprüfung der Benutzereingabe im Beschreibungsfeld angezeigt. Dieses Feld stammt ebenfalls aus der Auswahltabelle.

Return Field Name (Code) – Der Name des Feldes mit dem vom Anwender eingegebenen Wert aus der Auswahltabelle. In der Regel entspricht dieser Feldname dem Namen, der in *txtField.ControlSource* angegeben ist. Hier ist nur der Feldname ohne den Tabellennamen anzugeben. Der Wert dieses Feldes muss vom Typ „Zeichen“ sein. Gegebenenfalls ist der Wert mit TRANSFORM() in einen Zeichentyp umzuwandeln.

Return Field Name (Description) – Der Name des Feldes mit der Beschreibung, die aus der Auswahltabelle zurückgegeben wird. Es kann auch ein Ausdruck zurückgegeben werden. Der Wert wird im Beschreibungsfeld angezeigt. Der Wert muss vom Typ „Zeichen“ sein. Gegebenenfalls ist der Wert mit TRANSFORM() in einen Zeichentyp umzuwandeln.

Return Field Name (Internal Key) – Der Name des Feldes aus der Auswahltabelle, das den Primärschlüssel enthält. Über dieses Feld wird die Beziehung von der Bearbeitungstabelle zur Auswahltabelle in der Datenumgebung hergestellt.

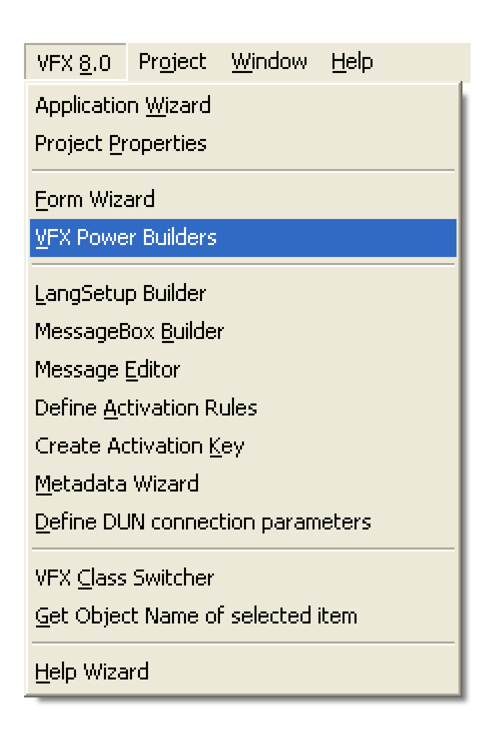
Control Source Internal Key – Der Name des Feldes aus der Bearbeitungstabelle, das den Primärschlüssel enthält. Dieses Feld enthält den Fremdschlüssel aus der Auswahltabelle.

8.14. Der VFX – CPickTextBox Builder

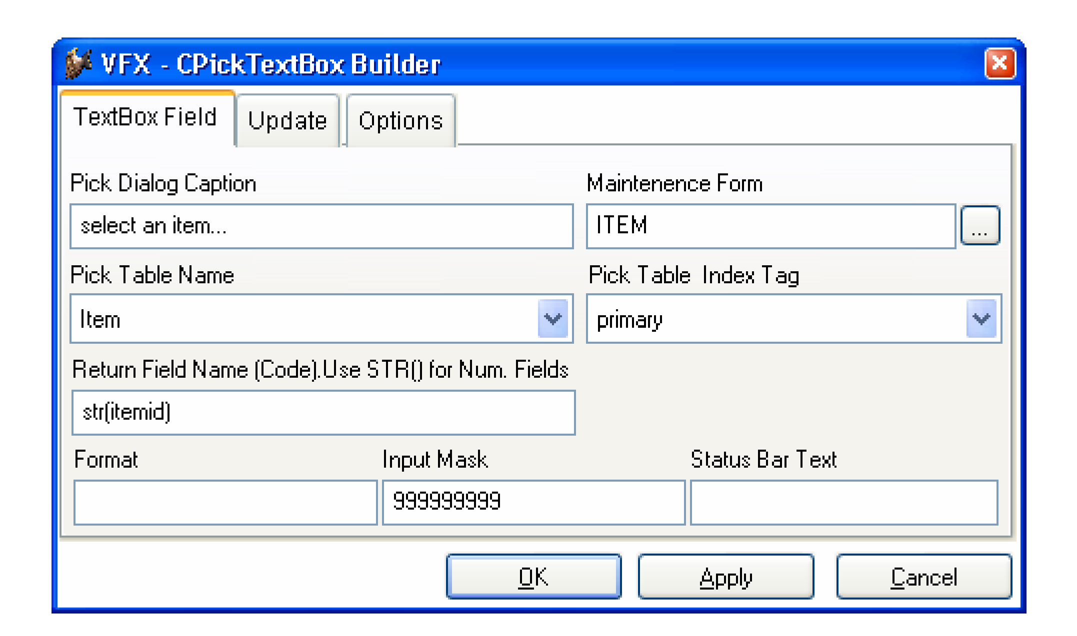
Visual Extend bietet einen Builder, um leistungsfähige Auswahltextfelder zu erstellen. Die Auswahltextfelder können in Child-Grids verwendet werden.

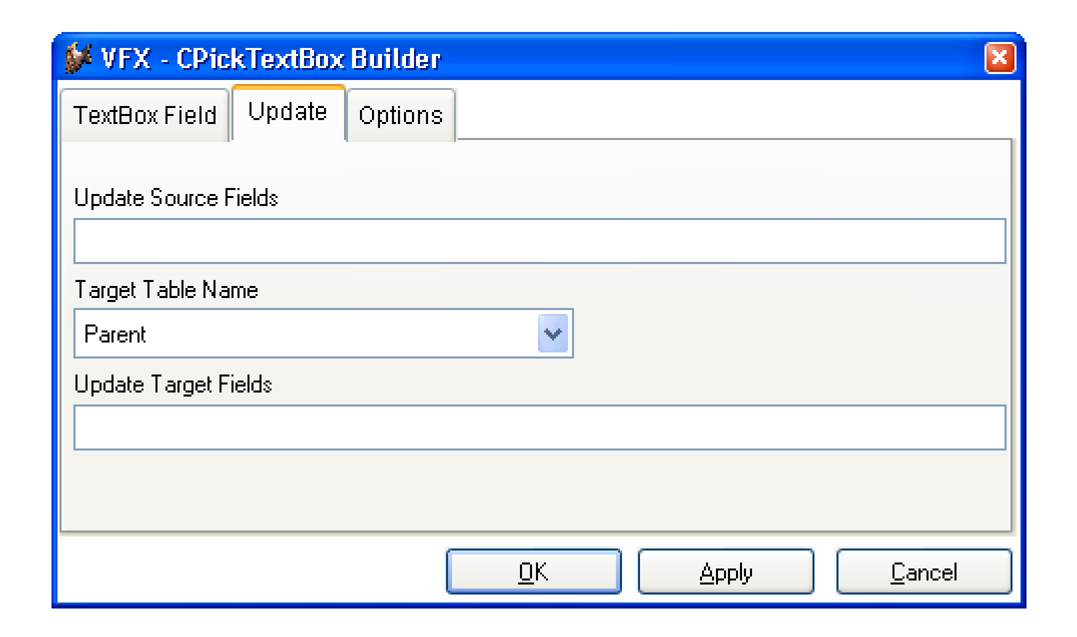
8.14.1. Aufruf des VFX – CPickTextBox Builder

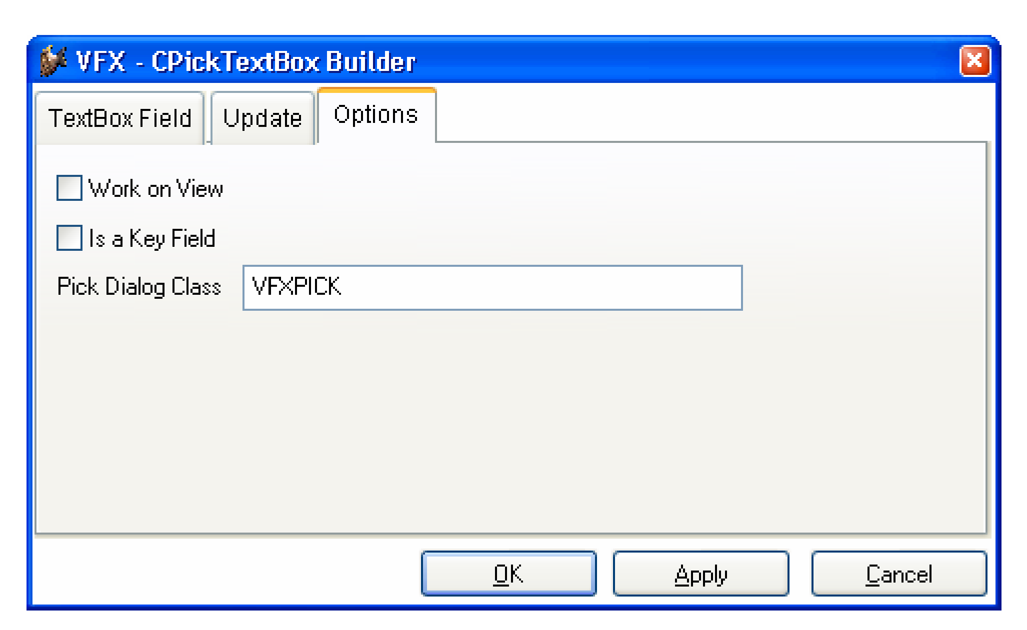
Um den VFX- CPickTextBox Builder aufzurufen, wählen Sie die Spalte im Grid, die das Auswahltextfeld erhalten soll und wählen Sie den Menüpunkt VFX Power Builder aus dem VFX-Menü:



Der VFX - CPickTextBox Builder ist in der Bedienung dem normalen VFX – CPickField Builder ähnlich und ist ebenfalls voll wieder verwendbar:







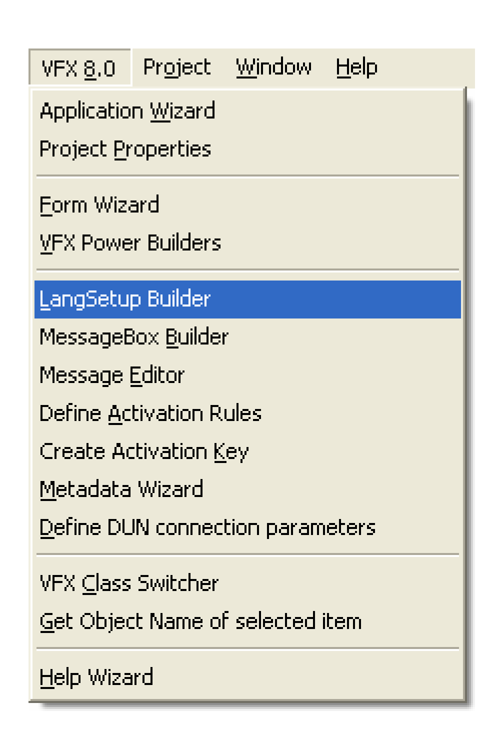
8.15. Der VFX – LangSetup Builder

8.15.1. Ziel

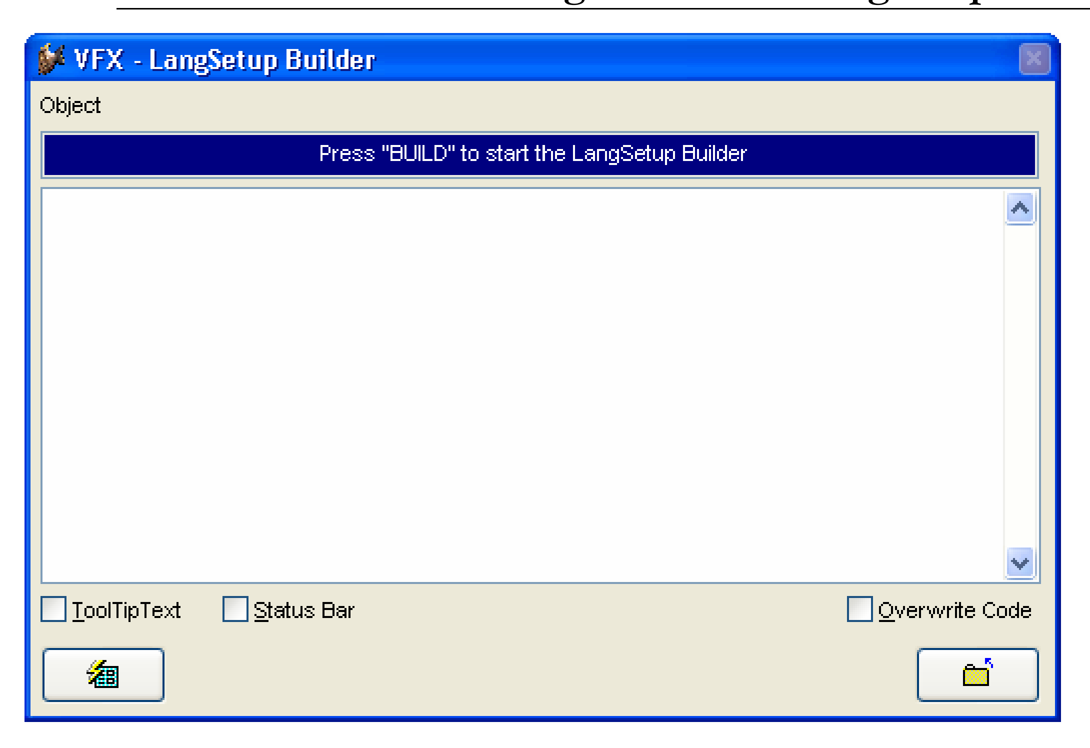
Der VFX – LangSetup Builder automatisiert die Erstellung des in der *LangSetup()*-Methode benötigten Codes. Sie brauchen diesen Code, wenn Sie Ihre Anwendung in mehr als einer Sprache erstellen wollen. Das Ziel dieses Builders ist es, aus dem Formular für alle Bezeichnungen, Tooltip-Texte und Statuszeilenmeldungen Konstanten anzulegen und diese in der Tabelle *VFXMSG.DBF* zu speichern. Nach diesem Vorgang können Sie den VFX – Message Editor, weiter unten in diesem Handbuch beschrieben, benutzen, um die Texte in verschiedene Sprachen zu übersetzen.

8.15.2. Aufruf des VFX – LangSetup Builders

Um den VFX – LangSetup Builder aufzurufen, öffnen Sie zunächst das Formular dessen Bezeichnungen, Tooltip-Texte und Statuszeilenmeldungen Sie analysieren lassen möchten. Wir könnten sagen alle für die Übersetzung in Frage kommenden Texte. Wählen Sie den folgenden Menüpunkt aus dem VFX-Menü:

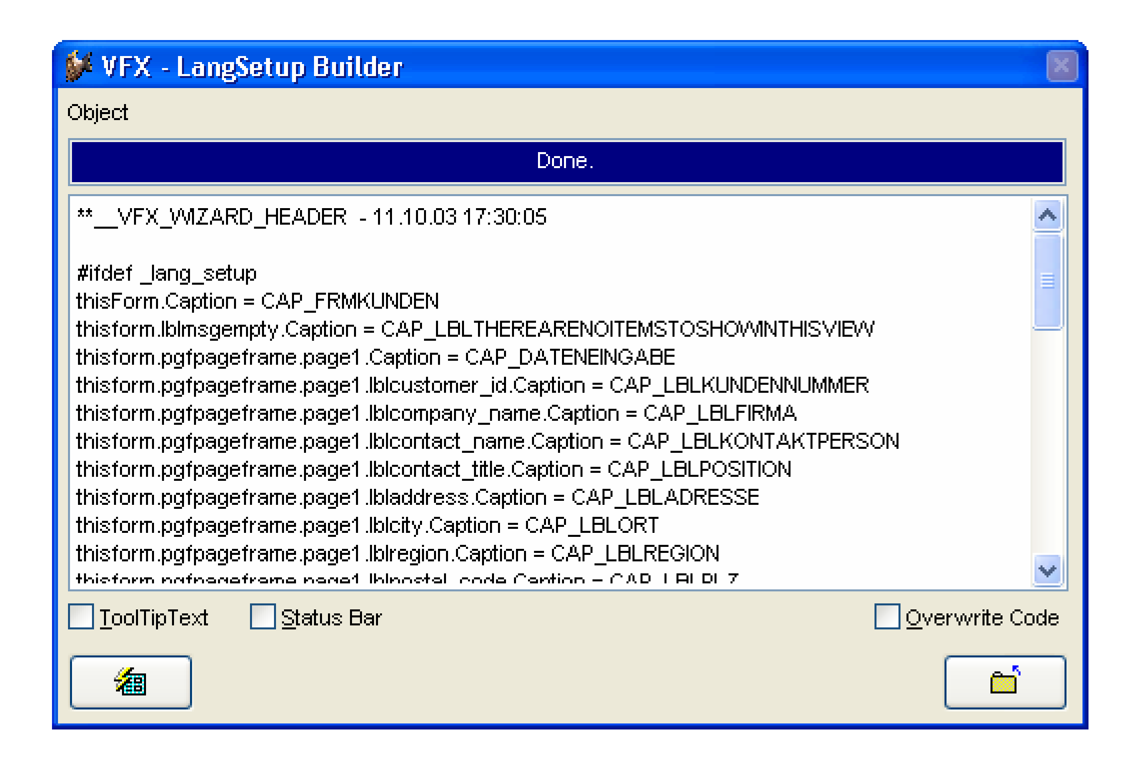


8.15.3. Die Bedienung des VFX – LangSetup Builders



Markieren Sie die Kontrollkästchen entsprechend den gewünschten Optionen. Klicken Sie auf die Schaltfläche Build um den Code für die *LangSetup()*-Methode generieren zu lassen.

Nach der Generierung sehen Sie den Code, der für die *LangSetup()*-Methode erzeugt wurde. Wenn Sie das Kontrollkästchen *Overwrite Code* markieren, wird der erzeugte Code in die *LangSetup()*-Methode des aktuell in der Entwicklungsansicht geöffneten Formulars geschrieben. Der Bezeichnungscode wird in der VFX-Meldungstabelle *VFXMSG.DBF* gespeichert. Hier können Sie die Texte bearbeiten und in andere Sprachen übersetzen.



Beachten Sie, dass die Konstanten automatisch in die Tabelle *VFXMSG.DBF* eingefügt werden, wenn Sie das Kontrollkästchen *Overwrite Code* markieren.

8.15.4. Define _Lang_Setup

In der Include-Datei *VFX.H* gibt die _LANG_SETUP-Konstante an, ob die *LangSetup()*-Methode ausgeführt wird. In der *LangSetup()*-Methode wird überprüft, ob diese Konstante existiert und falls ja, wird der Code der Methode ausgeführt. Dieses Verfahren dient der Geschwindigkeitsoptimierung für die Formulare.

```
#DEFINE _LANG_SETUP           .T.
```

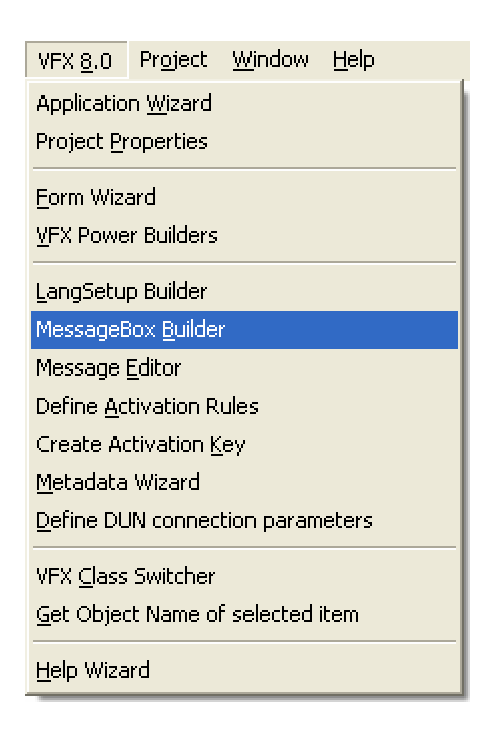
8.16. Der VFX – Messagebox Builder

8.16.1. Ziel

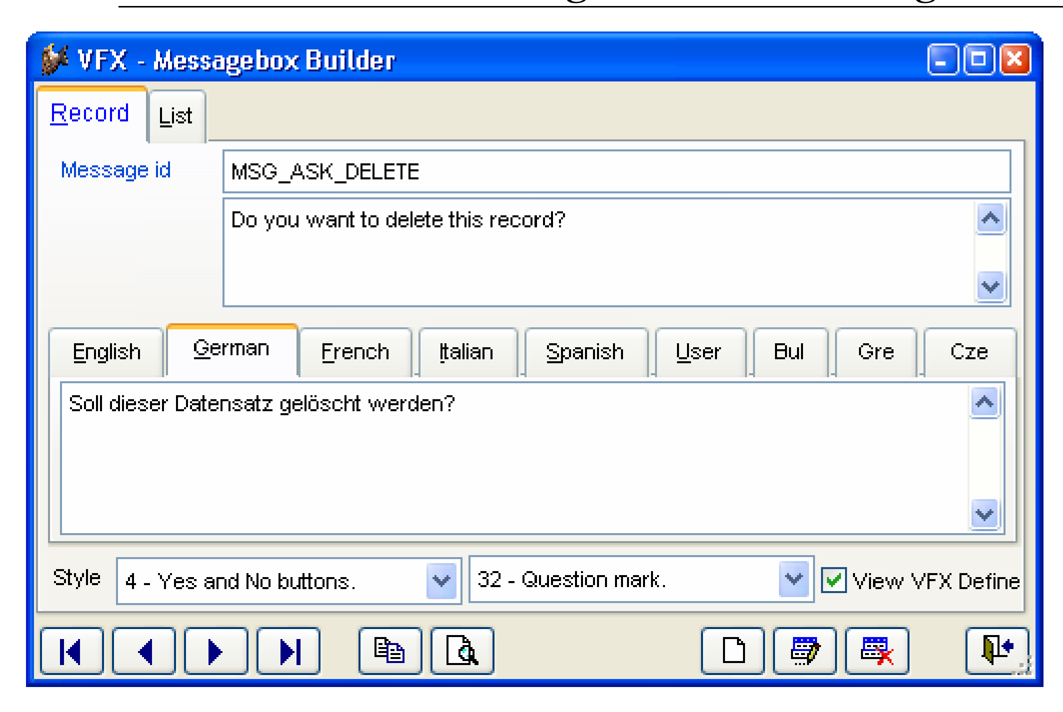
Der VFX – Messagebox Builder ist ein praktisches Werkzeug, um Meldungsdialoge (und WAIT WINDOWs) während der Entwicklung zu erstellen. Der VFX – Messagebox Builder hilft Ihnen nicht nur bei der Erstellung der Codezeile für den Meldungsdialog, sondern trägt die Texte auch gleich in die Tabelle *VFXMSG.DBF* ein, wo Sie diese weiter bearbeiten und in andere Sprachen übersetzen können. Der VFX – Message Editor wird weiter unten in diesem Handbuch beschrieben.

8.16.2. Aufruf des VFX – Messagebox Builder

Um den VFX – Messagebox Builder aufzurufen, wählen Sie den folgenden Menüpunkt aus dem VFX-Menü:



8.16.3. Die Bedienung des VFX – Messagebox Builder



Klicken Sie auf die Schaltfläche *neu* um eine neue Messagebox anzulegen. Tragen Sie dann im Feld *Message id* eine eindeutige Bezeichnung für die Messagebox ein. Im Seitenrahmen können Sie für jede benötigte Sprache den Text hinterlegen.

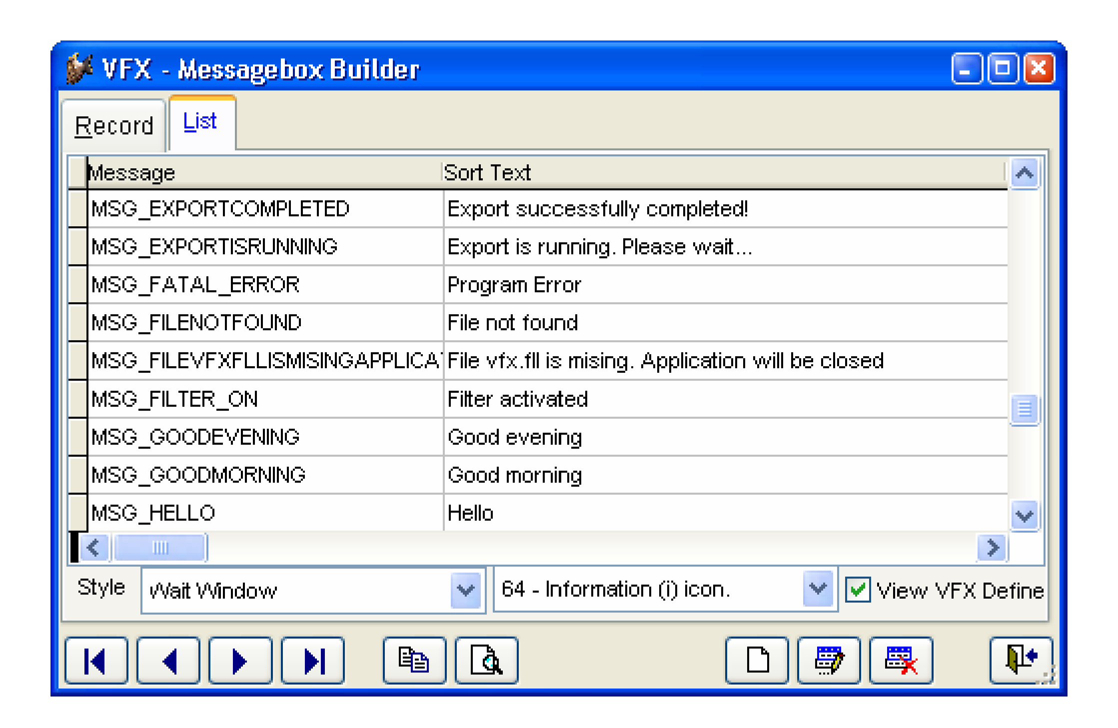
In der Zeile *Style* wählen Sie gewünschten Typ der Messagebox aus. Es kann zwischen verschiedenen Symbolen und Schaltflächen auf der Messagebox ausgewählt werden.

Durch einen Klick auf die Schaltfläche *Test it!* wird die Messagebox in der Vorschau angezeigt.

Kopieren Sie den vom VFX – Messagebox Builder erstellten Code mit der Schaltfläche *Copy code to clipboard* in die Zwischenablage. Aus der Zwischenablage kann der Code in einem beliebigen Programmteil eingefügt werden.

Der VFX – Messagebox Builder legt für jeden Eintrag einen Datensatz in der Tabelle *VFXMSG.DBF* an.

Auf der Seite *List* erhalten Sie eine Übersicht über alle vorhandenen Datensätze:



Tipp: Auch wenn Sie keine mehrsprachigen Anwendungen erstellen, können Sie den VFX – Messagebox Builder einsetzen.

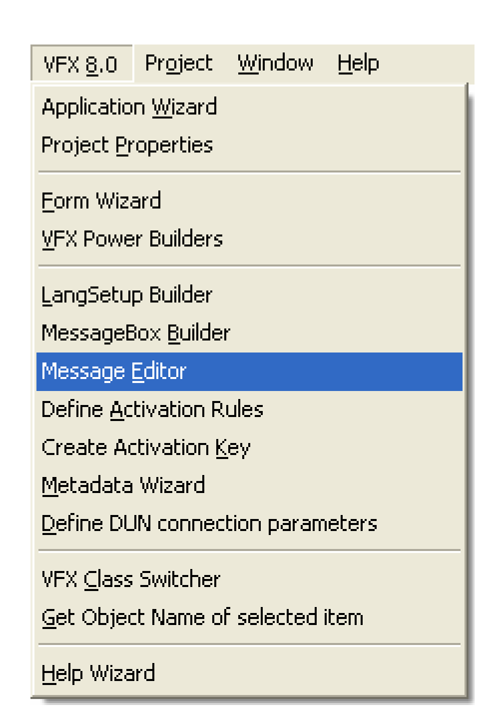
8.17. Der VFX – Message Editor

8.17.1. Ziel

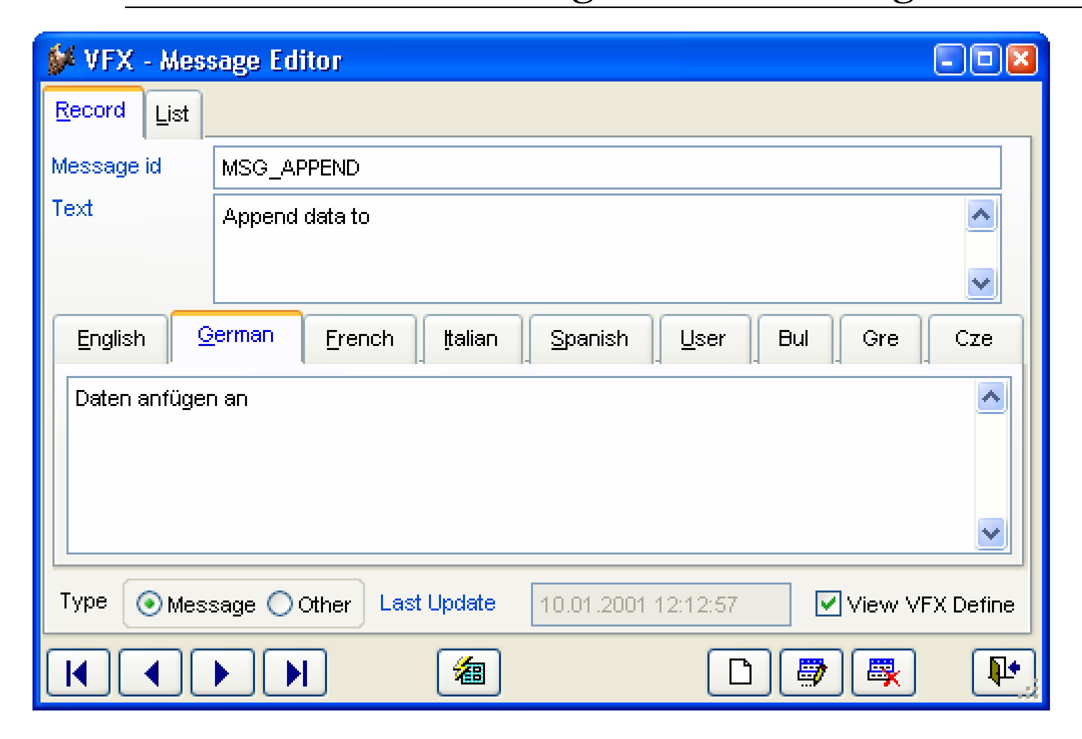
Der VFX – Message Editor ist der Zentrale Ort um, alle Bezeichnungen, Meldungen, Tooltip-Texte und Statuszeilenmeldungen zu verwalten und in andere Sprachen zu übersetzen. Aus dem VFX – Message Editor heraus können Sie die benötigten Include-Dateien (*USERTEXT.H* und *USERMSG.H*) erstellen.

8.17.2. Aufruf des VFX – Message Editor

Um den VFX – Message Editor aufzurufen, wählen Sie den folgenden Menüpunkt aus dem VFX-Menü:



8.17.3. Die Bedienung des VFX-Message-Editor



Klicken Sie auf die Schaltfläche *Make Include File* um eine Include-Datei in der im Seitenrahmen angezeigten Sprache zu erstellen. Die Include-Dateien werden in einem Ordner mit der Bezeichnung der jeweiligen Sprache unterhalb des Include-Ordners Ihres Projektes gespeichert.

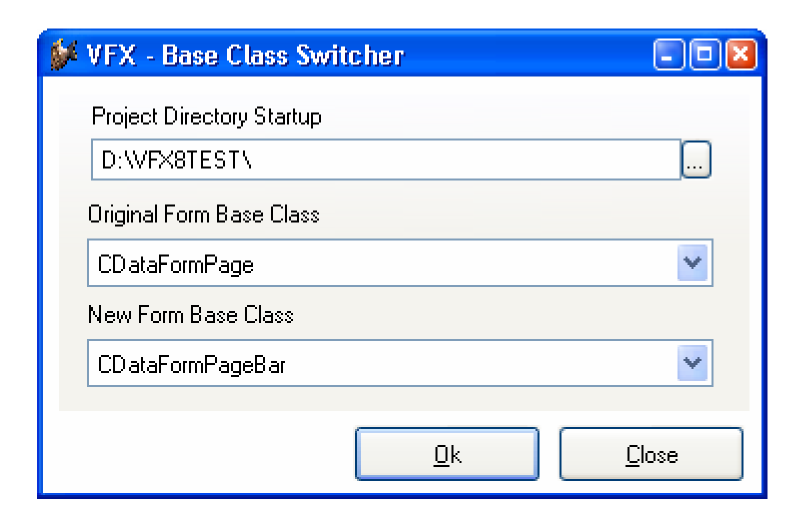
Nach der Erstellung Ihrer Include-Dateien müssen Sie diese nur noch in den *INCLUDE*-Ordner Ihres Projektes kopieren, wie im Kapitel *Erstellen mehrsprachiger Anwendungen* beschrieben ist. **Tipp:** Sie können Ihre eigenen Kon-

stanten mit den erzeugten Konstanten aus der Tabelle *VFXMSG.DBF* mischen. Schreiben Sie Ihre Konstanten vor oder nach dem VFX-Header bzw. -Footer.

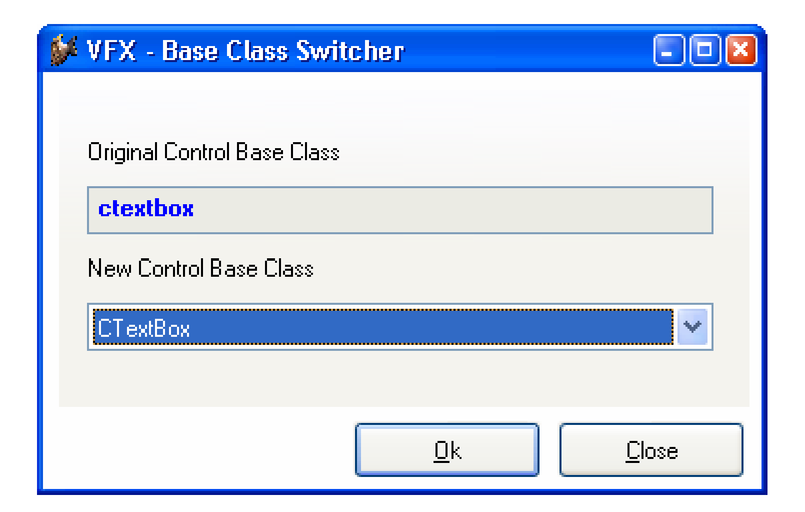
8.18. Der VFX – Class Switcher

Der Class Switcher hat zwei Funktionen.

Wenn beim Aufruf kein Formular geöffnet ist, ändert der Class Switcher die Klassen von Formularen in einem ganzen Projekt. Zum Beispiel kann die Formularklasse CDataFormPageBar durch CDataFormPage ersetzt werden. Dadurch ist es möglich alle Formulare mit Schaltflächen auszustatten bzw. diese wieder zu entfernen. Als besonders hilfreich erweist sich dieses Werkzeug bei der Aktualisierung vorhandener VFX 3-Projekte. In VFX 3 hatte jedes Formular am unteren Rand eine Leiste mit Schaltflächen. In VFX 8.0 kann man stattdessen eine richtige Symbolleiste verwenden.

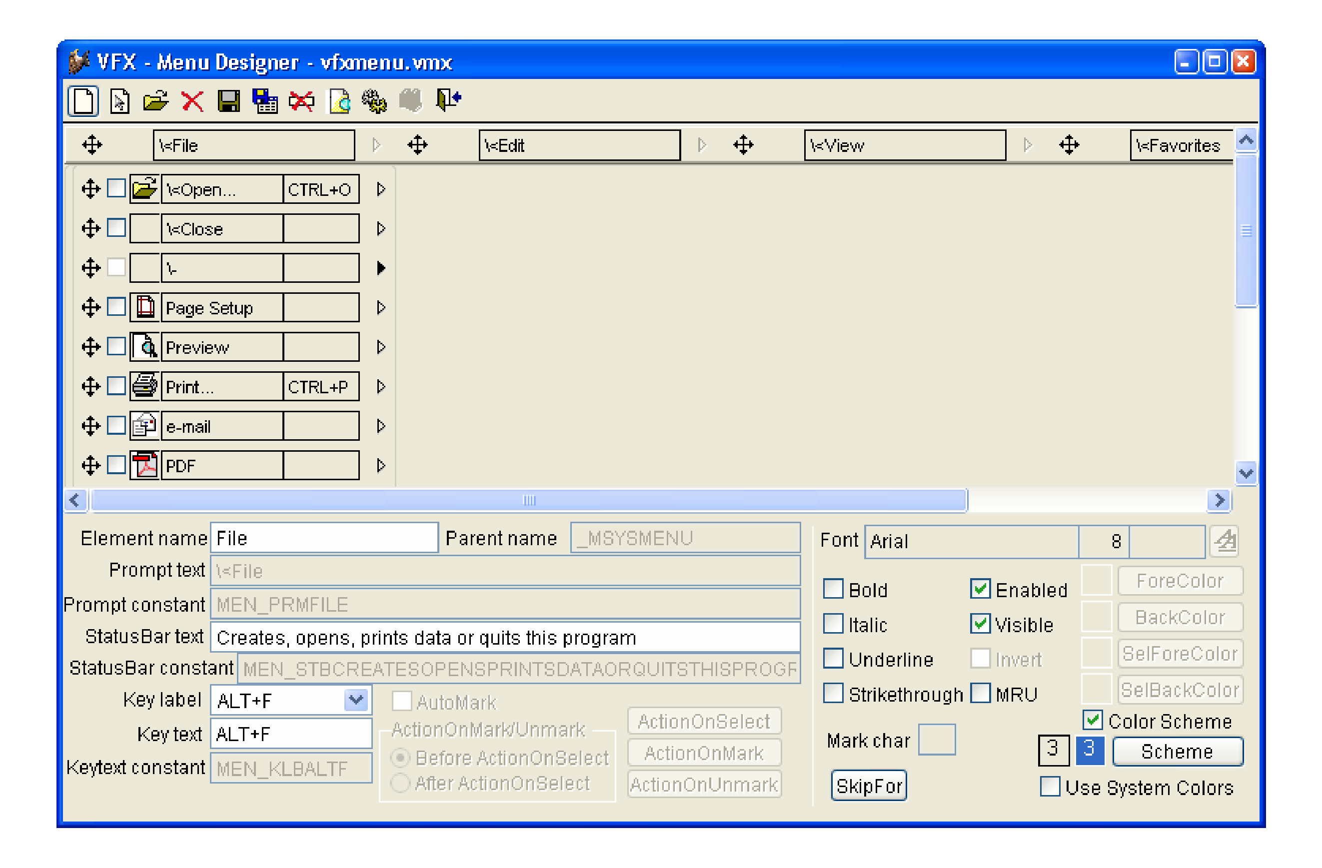


Wenn beim Aufruf des VFX - Class Switcher ein Formular zur Bearbeitung geöffnet ist, können die einzelnen Objekten zugrunde liegenden Klassen geändert werden. So ist es z. B. möglich, aus einer Textbox nachträglich ein Drehfeld zu machen.



8.19. Der VFX Menü-Designer

Der VFX Menü-Designer (VMD) ist ein Werkzeug zur schnellen Entwicklung von Menüs. Der VMD ist ein visueller Designer, in dem das Menü schon während der Entwicklung so angezeigt wird, wie es zur Laufzeit aussehen wird. Der VMD macht die Entwicklung einfacher und ermöglicht die schnelle Einstellung aller Menü-Eigenschaften im Gegensatz vom VFP Menü-Designer, der nicht alle Eigenschaften von Menüs unterstützt. Es können mehrsprachige Menüs erstellt werden, indem auf die entsprechende Schaltfläche in der Symbolleiste geklickt wird.



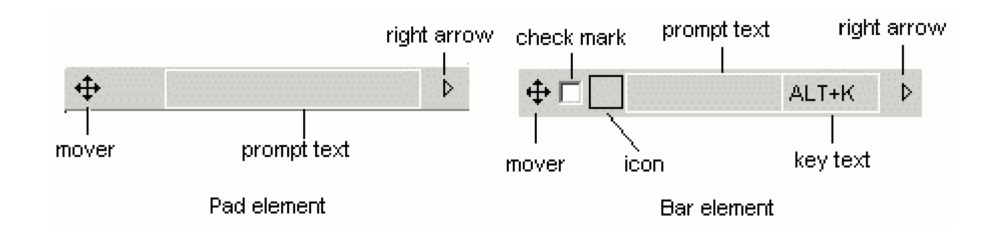
Ein in einem VFX-Projekt enthaltenes Menü kann direkt aus dem VFP-Projekt-Manager mit dem VMD geöffnet werden. Wahlweise können Menüs auch aus dem VMD heraus über das Öffnen-Symbol in der Symbolleiste oder über den entsprechenden Menüpunkt geöffnet werden. Im Öffnen-Dialog kann zwischen den Menütypen .mnx und .vmx gewechselt werden. Wenn ein Menü geöffnet wird, das noch nie mit dem VMD bearbeitet wurde, wird es automatisch in das .vmx-Format konvertiert.

Das geöffnete Menü kann visuell bearbeitet werden. Es können Einträge hinzugefügt und gelöscht werden und es können die Eigenschaften der einzelnen Einträge bearbeitet werden.

Neue Menü-Pads können durch einen Klick auf den Rechtspfeil, der sich rechts neben jedem Menüeintrag befindet, angelegt werden. Dadurch wird ein Untermenü angelegt. Einem Menü können neue Einträge hinzugefügt werden, indem auf den Pfeil nach unten unterhalb des Eintrags geklickt wird.

Ein Menüeintrag oder ein Menü-Pad können gelöscht werden, wenn sich der Fokus darauf befindet. Über den Menüpunkt löschen oder mit der Tastenkombination Strg+Entf wird der markierte Eintrag gelöscht.

Einige der Eigenschaften eines Menüeintrags können visuell eingestellt werden:



- Prompt text

Der angezeigte Text kann direkt eingetragen werden, wenn sich der Fokus auf dem jeweiligen Eintrag befindet. Die Textbox „Prompt text" im unteren Teil des VMD dient nur zur Anzeige des aktiven Eintrags im visuellen Teil des Designers.

- Key text

Die Bezeichnung des Tastenschlüssels zeigt dem Anwender die Zugriffstaste oder Tastenkombination an, mit der der Eintrag ausgewählt werden kann. Die Bezeichnung sollte dem im unteren Teil des VMD gewählten Tastenschlüssels entsprechen.

- Check mark

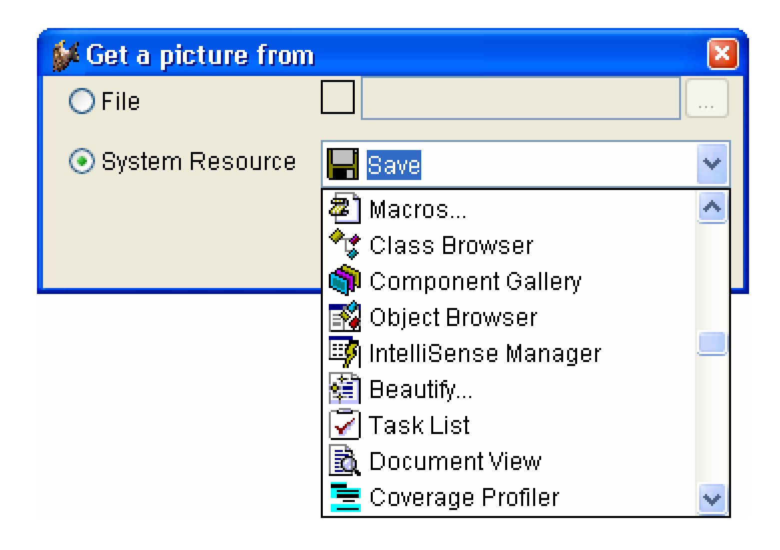
Damit sich ein Menüeintrag wie ein Kontrollkästchen verhält, muss bei „*AutoMark*" eine Markierung gesetzt werden. Wenn zusätzlich eine Markierung

bei „*Check mark*“ gesetzt wird, ist der Menüeintrag bereits beim Laden des Menüs markiert. Für Menüeinträge, die sich wie ein Kontrollkästchen verhalten, können zusätzliche Code-Teile ausgeführt werden, wenn das entsprechende Kontrollkästchen markiert wird („*ActionOnMark*“) bzw. wenn die Markierung aufgehoben wird („*ActionOnUnmark*“). Im Standard-VFP-Editorfenster kann der jeweilige Code bearbeitet werden.

Der Code, der bei *ActionOnMark* oder *ActionOnUnmark* eingegeben wird, kann wahlweise vor oder nach der *ActionOnSelect* ausgeführt werden. Um dieses Verhalten einzustellen, ist die entsprechende Option „*Before ActionOnSelect*“ oder „*After ActionOnSelect*“ auszuwählen.

- *Icon*

Jedem Eintrag in einem Menü kann ein Symbol zugeordnet werden. Dieses Symbol kann aus den in VFP integrierten Systemressourcen ausgewählt werden oder es kann eine Datei verwendet werden. Durch einen Klick auf das schwarzumrandete Kästchen kann ein Symbol mithilfe des „*Get a picture from*“-Dialogs ausgewählt werden. In diesem Dialog kann zwischen einer Datei und einem Symbol aus den VFP Systemressourcen gewählt werden.



Wenn einem Menüeintrag ein Symbol zugeordnet ist und sich dieser Menüeintrag wie ein Kontrollkästchen verhalten soll, dient das Symbol als Markierung. Wenn der Eintrag markiert wird, erscheint das Symbol eingedrückt. Wenn die Markierung aufgehoben wird, erscheint das Symbol normal.

Die Position der einzelnen Einträge innerhalb des Menüs kann per drag & drop verändert werden. Für diesen Vorgang ist der Vierwegepfeil ✥, der sich links neben allen Einträgen befindet festzuhalten. In einigen Fällen sind Verschiebeoperationen nicht möglich. Ein Menü-Pad kann nicht in einen Menüeintrag umgewandelt werden und umgekehrt. Außerdem ist es nicht möglich einen Menüeintrag in ein Untermenü zu verschieben.

Weitere Eigenschaften der Menüeinträge können im unteren Teil des Menü-Designers eingestellt werden. Dazu gehören der Zeichensatz, die Vordergrund- und Hintergrundfarbe, eine Meldung, die in der VFP-Statusbar angezeigt wird sowie der Name einer Konstanten, die verwendet wird, wenn ein mehrsprachiges Menü erstellt wird. Alle Änderungen werden unmittelbar im aktiven Element angezeigt.

Mit der Schaltfläche „*ActionOnSelect*“ kann in einem Editor-Fenster die auszuführende Aktion eingegeben werden. Über die Schaltfläche „*SkipFor*“ kann eine Bedingung eingegeben werden. Wenn diese Bedingung .T. liefert, kann der dazugehörige Menüeintrag nicht ausgewählt werden.

Die eingestellten Eigenschaften beziehen sich immer auf den aktiven Menüeintrag. Neue Menüeinträge erben die Eigenschaften des zuvor ausgewählten Eintrags.

Der Zeichensatz kann über die Schaltfläche „*Font*“ ausgewählt werden. Der Standard-Windows-Dialog zur Auswahl eines Zeichensatzes erscheint. In diesem Dialog können insbesondere die Schriftart und die Schriftgröße sowie der Schriftschnitt ausgewählt werden.

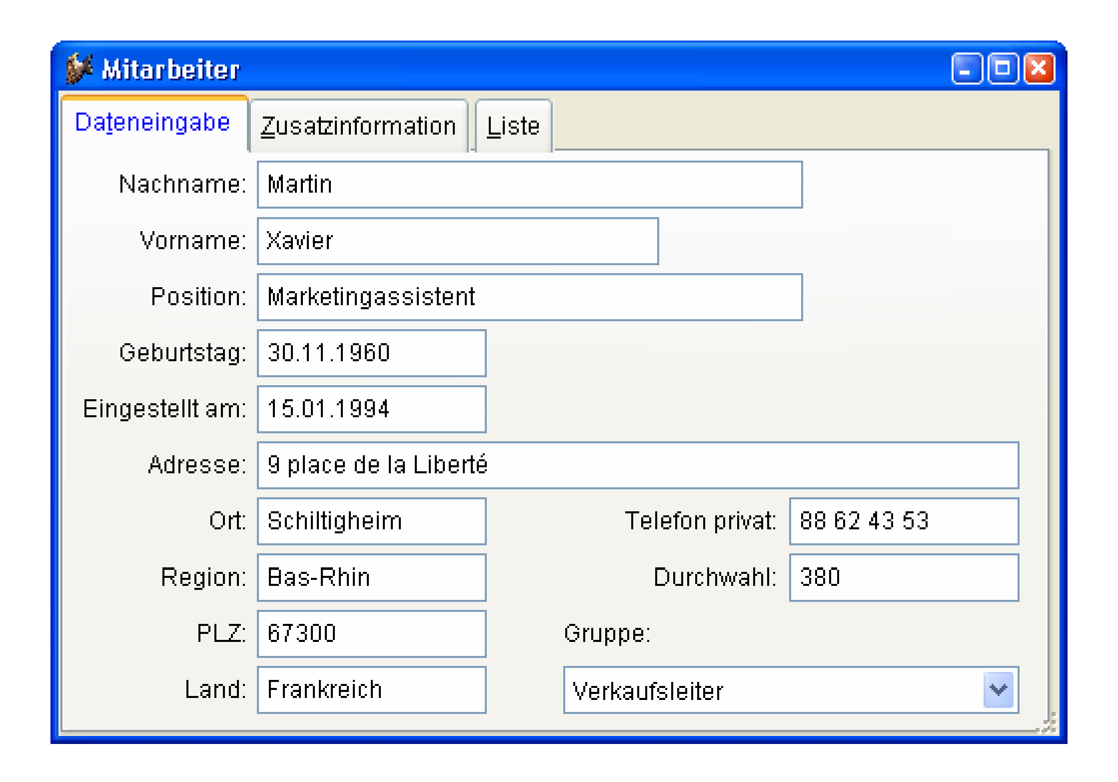
Zu jeder Zeit kann eine Vorschau des Menüs angezeigt werden, indem in der Symbolleiste oder im VMD-Menü „*Preview*“ gewählt wird.

9. Eigenschaften der erstellten Formulare

Die mit den VFX – Form Buildern erstellten Formulare haben standardmäßig viele gute Eigenschaften. Die Position des Formulars auf dem Bildschirm, die Größe des Formulars (die Größe eines Formulars kann mithilfe eines Resizers vom Benutzer zur Laufzeit eingestellt werden), die zuletzt aktive Seite des Seitenrahmens sowie die Einstellungen des Datenrasters, Sortierfolge, Spaltenbreiten, werden für jeden Benutzer individuell gespeichert. Schließt ein Benutzer ein Formular und öffnet er es wieder, erscheint es genauso, wie er es verlassen hat.

9.1. Formularbedienung

Die Standardbedienung für ein Standard-Datenbearbeitungsformular sieht wie folgt aus, wenn Sie sich nicht im Bearbeitungsmodus oder im Einfügemodus befinden:



Wenn Sie sich im Einfüge- oder Bearbeitungsmodus befinden, ändert sich die Überschrift des Formulars und die Schaltflächen der Symbolleiste werden entsprechend aktualisiert.

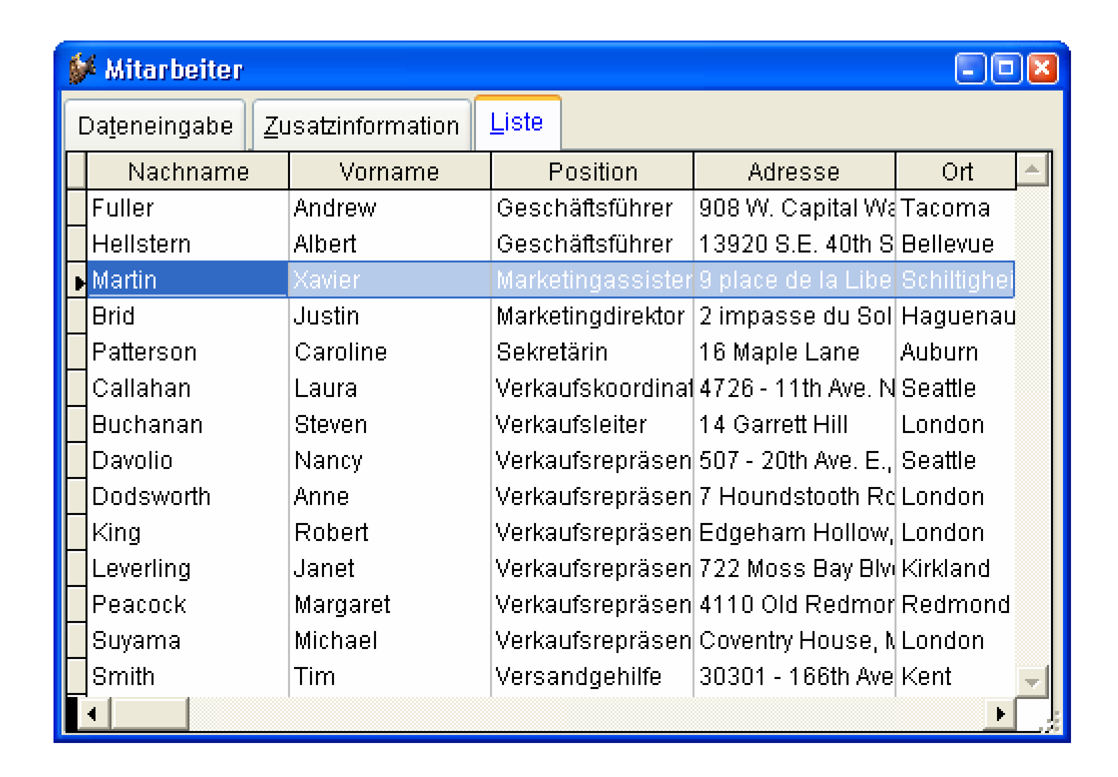
ANMERKUNG: Um große Datenmengen einzugeben, können Sie die Tastenkombination *Strg+N* drücken, auch wenn Sie sich bereits im Einfügemodus befinden. Dadurch ist es sehr schnell, mehrere Datensätze nacheinander zu erfassen. Aus den gleichen Optimierungsgründen bleiben die Navigations-Schaltflächen auch während der Bearbeitung aktiv.

Entsprechend der Einstellung der Eigenschaft *nAutoEdit* im Anwendungsobjekt bzw. der Formulareigenschaft *lAutoEdit* kann der Benutzer einfach mit der Bearbeitung beginnen und das Formular wechselt automatisch in den Bearbeitungsmodus, wie hier gezeigt wird:

Die Schaltflächen der Symbolleiste sowie die Menüeinträge werden entsprechend dem Formularstatus aktiviert.

9.2. Das VFX Power Grid

In allen Spalten eines Grids ist standardmäßig eine inkrementelle Suche möglich. Durch einen Doppelklick auf eine Überschrift in einem Datenraster kann die entsprechende Spalte sortiert werden. Wenn für die Spalte kein geeigneter Index vorhanden ist, wird von VFX automatisch ein temporärer Index angelegt. Soll die Suche um eine zusätzliche Spalte erweitert werden, drückt man die Taste „Strg" und klickt gleichzeitig auf eine weitere Überschrift. Die Rangfolge der Sortierung wird in den Überschriften durch Zahlen in Klammern dargestellt.



Ein Doppelklick auf eine Überschrift sortiert eine Spalte. Ein weiterer Doppelklick kehrt die Sortierfolge um.

Nach einem Klick in eine Spalte kann mit der Eingabe eines Suchbegriffs begonnen werden. Die Sortierfolge wird auf diese Spalte umgestellt und der eingegebene Begriff wird inkrementell gesucht. Der eingegebene Begriff wird in der Statuszeile angezeigt:

Suche : Martin

9.2.1. Inkrementelle Suche

Benutzen Sie den VFX CGrid Builder, um einzustellen für welche Spalten die inkrementelle Suche verwendet werden soll. Dadurch erhält der Benutzer die Möglichkeit, durch einfaches Eingeben eines Zeichens, einer Zahl oder auch eines Datums die inkrementelle Suche einzuleiten. Dabei wird die Sortierfolge automatisch umgestellt und es wird auf den der Eingabe entsprechenden Eintrag gesprungen. Während der inkrementellen Suche, wird der Suchbegriff in der Statuszeile angezeigt. Korrekturen können mit der Rückschritttaste durchgeführt werden.

9.2.2. Ändern der Sortierfolge durch Doppelklick auf eine Überschrift

Sortieren Sie eine Spalte durch einen Doppelklick auf die Spaltenüberschrift. Durch einen weiteren Doppelklick können Sie die Sortierfolge umkehren.

Wenn ein Indexschlüssel existiert, so wird dieser von VFX benutzt. Wenn kein Indexschlüssel existiert, erstellt VFX automatisch eine temporäre Indexdatei, die gelöscht wird, wenn das Formular geschlossen wird.

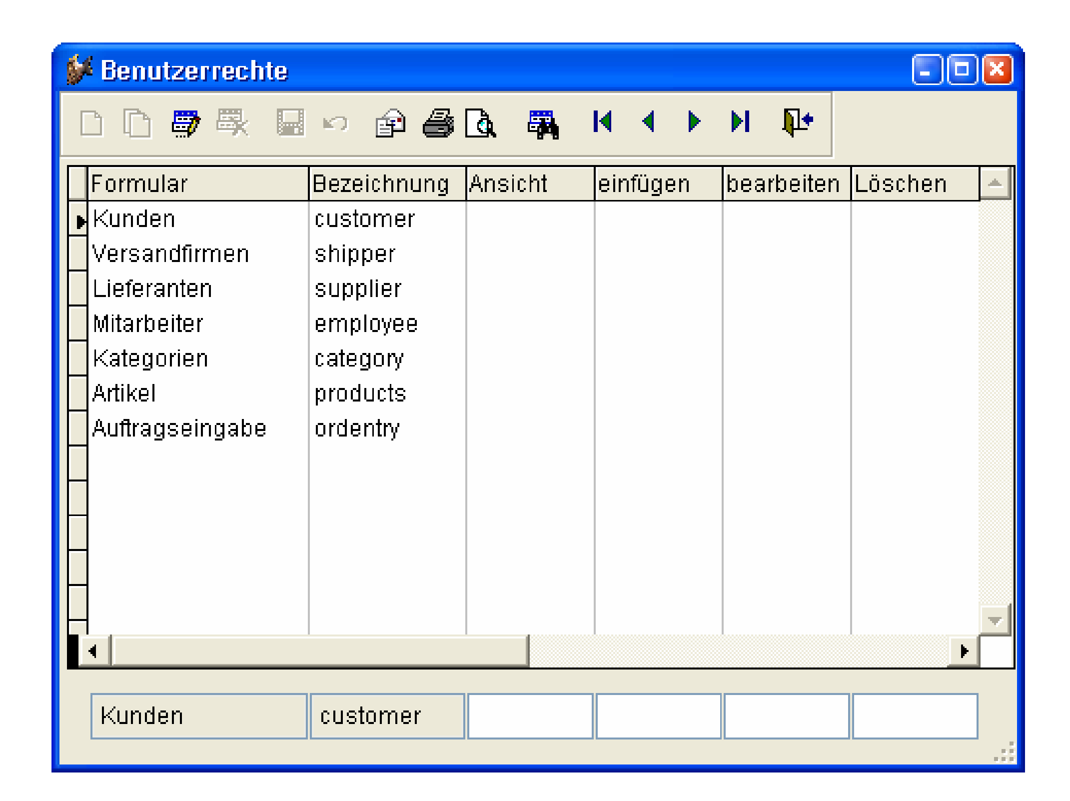
9.2.3. Anzeige der Sortierfolge in der Spaltenüberschrift

VFX zeigt die aktuelle Sortierfolge in der Spaltenüberschrift des Grids an. Der Entwickler kann aus den folgenden Anzeigemöglichkeiten auswählen:

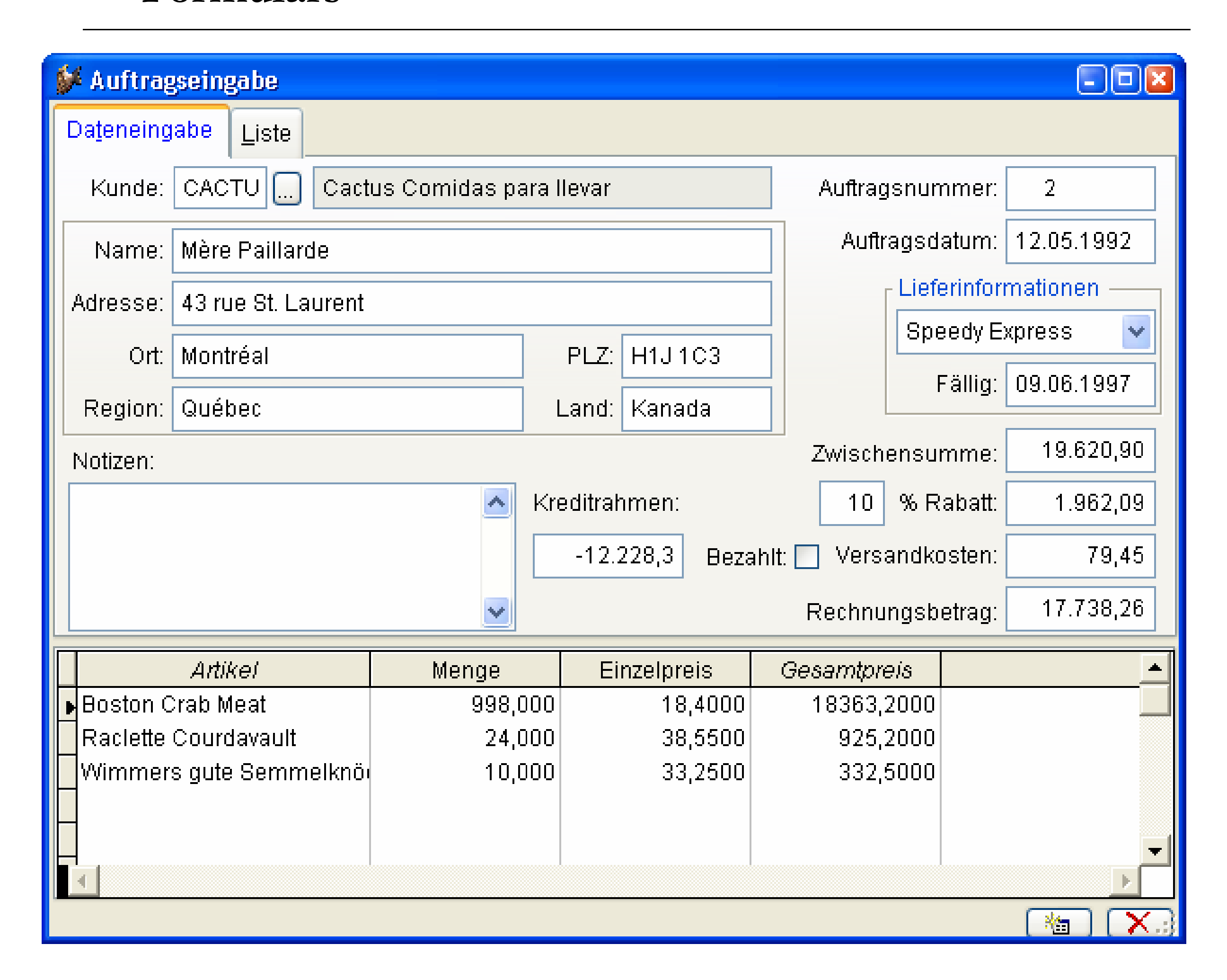
- Keine Anzeige
- Unterstrichene Überschrift
- Anzeige durch verschiedene Farben
- Anzeige durch einen auf- oder absteigenden Pfeil, ähnlich dem Windows-Explorer

9.3. Formulare basierend auf der Klasse CTableForm

Bei Formularen basierend auf der Klasse CTableForm sind das Such-Grid und andere Steuerelemente nebeneinander oder untereinander auf einem Container angeordnet. Ein typisches CTableForm-Formular ist die Verwaltung der Benutzerrechte.



9.4. Diskussion des VFX-1:n-Datenbearbeitungs-Formulars



9.4.1. Bearbeiten der Haupttabelle

Die Bearbeitung der Daten der Haupttabelle ist identisch mit der im Standard-Datenbearbeitungs-Formular. Die Symbolleiste und das Menü *Bearbeiten* beziehen sich auf die Haupttabelle.

9.4.2. Bearbeiten der Child-Tabelle

Die Child-Datensätze werden im unteren Grid bearbeitet. Nur wenn Sie sich im Bearbeitungs- oder Einfügemodus der Haupttabelle befinden, können Sie auch das Child-Grid bearbeiten, Child-Datensätze einfügen und löschen. Alle Bearbeitungen der Child-Datensätze werden mit optimistischer Tabellenpufferung durchgeführt. Wenn Sie sich entscheiden, Ihre Änderungen rückgängig zu machen, werden die Änderungen in allen Child-Datensätzen rückgängig gemacht. Wenn Sie sich entscheiden, die Änderungen zu speichern, werden

alle Änderungen an der Haupttabelle und in allen Child-Datensätzen gespeichert.

9.4.3. Auswahlliste innerhalb eines Child-Grids

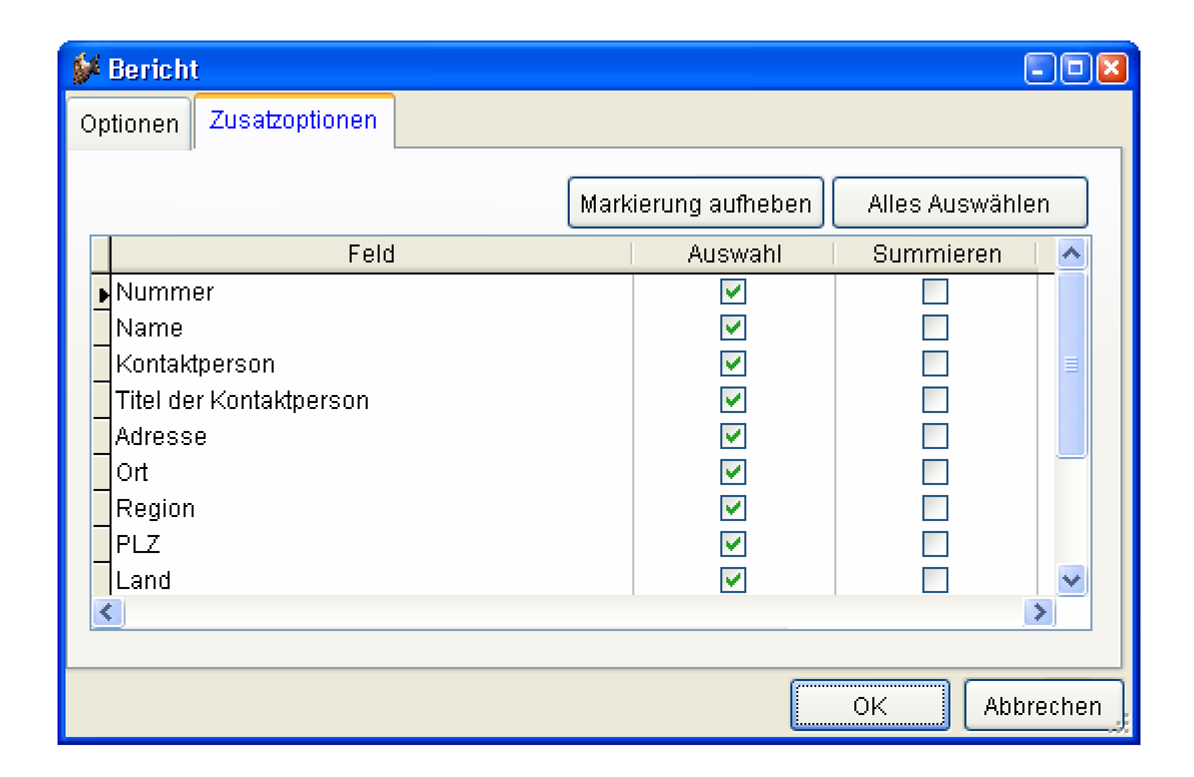
Eine der interessantesten Funktionen von VFX ist die besondere Auswahlliste, die Sie Ihrem Child-Grid mit dem VFX – CPickTextBox Builder hinzufügen können. Die Auswahllisten können im Bearbeitungs- und im Einfügemodus erreicht werden.

Durch einen Doppelklick in die *CPickTextBox* oder durch drücken der Funktionstaste *F9* wird die Auswahlliste angezeigt.

9.5. Drucken

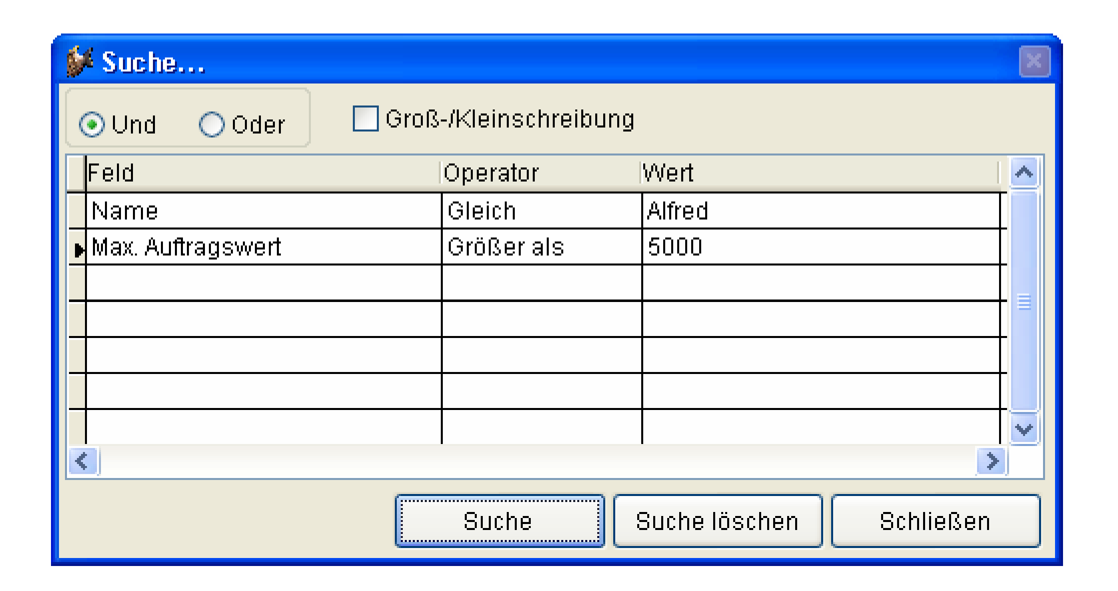
Aus allen Formularen kann standardmäßig eine Liste gedruckt werden, ohne dass dafür Berichte angelegt werden müssen. VFX legt zur Laufzeit der Anwendung temporäre Berichtsdateien an, die auf der Ansicht der Suchseite eines Formulars basieren.

Vor dem Druck bzw. der Seitenansicht kann der Benutzer nicht gewünschte Spalten aus der Liste entfernen. Die Breite der Spalten entspricht ungefähr der Breite der Spalte im Grid.



9.6. Filtern

Der sichtbare Datenbereich in einem Formular kann durch Setzen eines Filters eingeschränkt werden. VFX stellt dafür einen fertigen Dialog zur Verfügung. Beliebig viele Felder können dabei mit „und“ oder „oder“ verknüpft werden.



Es können jetzt beliebig viele Suchkriterien kombiniert werden. Die Suchkriterien werden je Benutzer und Formular gespeichert und stehen auch nach einem Neustart des Programms wieder zur Verfügung.

10. Applikationsschutz durch Produktaktivierung

Das Ziel der Produktaktivierung ist die unerlaubte Verwendung der Anwendung auf nicht aktivierten Computern zu verhindern.

Der Applikationsschutz durch Produktaktivierung kann im VFX – Application Wizard auf der Seite 3. Options durch aktivieren des Kontrollkästchens „Enable product activation" für ein neu zu erstellendes Projekt eingeschaltet werden.

Später kann diese Einstellung in Vfxmain.prg geändert werden. Die Eigenschaft goProgramm.lUseActivation muss auf .T. gesetzt werden, um die Produktaktivierung einzuschalten. Wenn die Eigenschaft goProgramm. lUseActivation auf .F. gesetzt ist, ist die Applikation nicht durch die Produktaktivierung geschützt.

Zu jeder Anwendung können bis zu 32 Rechte vergeben werden. Jedes Recht kann unabhängig von den anderen Rechten aktiviert werden.

10.1. Liste der verwendeten Begriffe

Systemspezifischer Wert – Ein systemspezifischer Wert, zum Beispiel die Seriennummer einer Hardware-Komponente oder das Erstellungsdatum einer bestimmten Datei oder ein Schlüssel aus der Windows-Registrierungsdatenbank. Die zu verwendete Datei und der zu verwendende Schlüssel aus der Windows-Registrierungsdatenbank können vom Entwickler festgelegt werden.

Aktivierungsregel – Für jede Applikation kann eine eindeutige Aktivierungsregel angelegt werden. Diese Regel setzt sich aus einer Reihe systemspezifischer Werte zusammen, die einen PC eindeutig identifizieren. Bei der Erstellung der Aktivierungsregel können Textbearbeitungsfunktionen verwendet werden.

Installationsschlüssel – Dies ist eine Zeichenkette, die Informationen über die im PC des Anwenders eingesetzte Hardware enthält. Der Installationsschlüssel wird vom Entwickler benötigt um einen Aktivierungsschlüssel erstellen zu können.

Aktivierungsschlüssel – Dies ist eine Zeichenkette, die die Berechtigungen für einen speziellen PC enthält. Der Aktivierungsschlüssel wird vom Ent-

wickler anhand des Installationsschlüssels erstellt. Der Aktivierungsschlüssel ist für andere PCs nutzlos.

Installationsdatum – An diesem Datum wurde eine Applikation erstmalig auf einem PC gestartet.

10.2. Das Funktionsprinzip

Wenn der Applikationsschutz durch Produktaktivierung aktiviert ist, wird beim Start der Applikation das Objekt *goProgram.SecurityRights* instanziiert. Dieses Objekt hat Eigenschaften mit den Namen der Benutzerrechte, die der Entwickler definiert hat. Jede dieser Eigenschaften kann einen von drei Werten annehmen:

-1 – Die Applikation ist nicht aktiviert. In diesem Fall kann der Entwickler entscheiden welche Aktion ausgeführt werden soll. Der Anwender könnte zum Beispiel begrenzten Zugriff auf Funktionen haben, solange die Applikation nicht aktiviert ist.

0 – Die Applikation ist aktiviert, aber der Anwender hat nicht das Recht diese Aktion auszuführen.

1 – Die Applikation ist aktiviert und der Anwender hat das Recht die Aktion auszuführen.

Wenn der Applikationsschutz durch Produktaktivierung aktiviert ist, werden der Aktivierungsschlüssel und das Datum des ersten Starts der Anwendung in einer Ini-Datei gespeichert. Der Entwickler kann den Namen dieser Ini-Datei selbst wählen, sodass jede Applikation ihre eigene Ini-Datei verwendet. Der Standardname ist *VFX.ini*. Die Ini-Datei wird im Windows-Ordner gespeichert.

Der Aktivierungsschlüssel wird durch die Aktivierungsregel verschlüsselt. Der Schutz kann durch Hinzufügen von Zeichenkonstanten, Schlüsseln aus der Windows-Registrierungsdatenbank und durch das Erstellungsdatum einer beliebigen Datei weiter verbessert werden. Diese Kombination kann für jede Applikation getrennt festgelegt werden, sodass jede Applikation ihre eigenen Aktivierungsregeln hat.

Zusätzlich zu diesen Einstellungen kann der Entwickler den Typ des Schutzes festlegen.

Der Standardschutz erstellt die Ini-Datei beim ersten Start der Applikation. Das während des Erstellens der INI-Datei aktuelle Systemdatum wird in der

Datei gespeichert. Dieses Datum steht während der Ausführung der Anwendung in der Eigenschaft *goProgram.InstallationDate* zur Verfügung und kann dazu verwendet werden die Laufzeit der Applikation zu beschränken. Der Nachteil dieses Schutzes ist, dass der Anwender die erstellte Ini-Datei löschen kann und die Ini-Datei beim nächsten Start der Applikation mit einem neuen Datum erneut erstellt wird.

Um eine solche Manipulation durch den Anwender auszuschließen, kann der Entwickler einen erweiterten Schutz einstellen. Hierbei wird eine zusätzliche Datei verwendet, die mit der Applikation vertrieben werden muss. Der Standardname dieser Datei heißt „FirstInstall.txt". Der Dateiname kann mit der Eigenschaft *cFirstInstall* aus der Klasse *cActivation* (appl.vcx) eingestellt werden. Die Datei „FirstInstall.txt" wird im Windows-Ordner abgelegt.

Wenn der Entwickler den Schutz mit der Datei „FirstInstall.txt" auswählt, wird sich die Applikation folgendermaßen verhalten. Beim Start der Applikation wird zunächst die Ini-Datei überprüft. Wenn diese Datei existiert, wird das Datum des ersten Starts der Eigenschaft *goProgram.InstallationDate* zugewiesen und die Benutzerrechte werden entsprechend dem Aktivierungsschlüssel eingestellt.

Wenn die Ini-Datei nicht existiert wird angenommen, dass dies der erste Start der Applikation ist. Wenn dies der Fall ist, wird zusätzlich überprüft, ob die Datei „FirstInstall.txt" existiert. Wenn diese Datei existiert ist sichergestellt, dass die Applikation wirklich zum ersten Mal gestartet wird. Das Systemdatum wird jetzt in der Ini-Datei gespeichert und die Datei „FirstInstall.txt" wird gelöscht. Wenn ein Anwender nun versucht eine Applikation zu reaktivieren indem er die Ini-Datei löscht, wird die Ausführung der Applikation beendet, weil die Datei „FirstInstall.txt" nicht existiert. Dieser erweiterte Schutz der Applikation bedeutet eine bessere Sicherheit. Der Entwickler darf jedoch nicht vergessen die Datei „FirstInstall.txt" beim Vertrieb der Applikation mit auszuliefern.

Wenn der Anwender die installierte Applikation aktivieren möchte, muss er seinen Installationsschlüssel an den Entwickler senden. Der Installationsschlüssel kann auf drei verschiedene Arten an den Entwickler gesendet werden. Die gewünschte Art kann in der Eigenschaft *nRegWay* eingestellt werden:

> 0 – Der Installationsschlüssel wird in einem Dialog angezeigt. Der Anwender kann den Schlüssel kopieren und in einer anderen Applikation (zum Beispiel in einer E-Mail) einfügen.
>
> 1 – Der Installationsschlüssel wird in einer Datei gespeichert. Diese Datei kann später an den Entwickler gesendet werden. Der Dateiname wird in der Eigenschaft cParamFile hinterlegt.

2 – Der Installationsschlüssel wird in einer Datei gespeichert und sofort als E-Mail-Anhang an den Entwickler geschickt. Der Dateiname muss in der Eigenschaft cParamFile hinterlegt werden. Die E-Mail-Adresse des Entwicklers muss in der Eigenschaft cRegEMail eingetragen werden.

Der Installationsschlüssel hat einen numerischen Wert mit 10 Stellen Länge. Der Anwender könnte den Installationsschlüssel per E-Mail an den Entwickler senden oder auf einer Registrierungs-Website eintragen. Der Entwickler trägt den Installationsschlüssel im „Create Activation Key"-Assistenten ein um einen Aktivierungsschlüssel für den Anwender zu erstellen. Der generierte Aktivierungsschlüssel wird dann an den Anwender geschickt und vom Anwender im Aktivierungsformular eingegeben um die Applikation zu aktivieren. Wahlweise kann die Datei mit dem Aktivierungsschlüssel auch einfach im Ordner der Exe-Datei gespeichert werden. Beim nächsten Start der Anwendung wird der Aktivierungsschlüssel aus dieser Datei gelesen.

Die Aktivierungsinformationen werden auf dem PC des Kunden in einer Ini-Datei gespeichert. Der Name dieser INI-Datei wird in der Eigenschaft *cINIFileName* der Klasse cVFXAcvtivation (Appl.vcx) eingetragen. Der Standardwert ist *VFX.ini*.

Der Entwickler kann wählen, ob die einfache Produktaktivierung verwendet werden soll oder ob zusätzlich die Datei „FirstInstall.txt" benutzt werden soll, um den ersten Start der Applikation zu protokollieren. Der Name dieser Datei kann in der Eigenschaft cFirstInstall der Klasse cVFXAcvtivation (Appl.vcx) eingetragen werden. Der Standardwert ist *FirstInstall.ini*.

Wenn die Datei „FirstInstall.txt" verwendet werden soll, muss diese Datei mit der Applikation vertrieben werden. Das Installationsprogramm muss diese Datei im Windows-Ordner speichern. Das Aktivierungsobjekt wird diese Datei beim ersten Start der Applikation löschen. In diesem Moment wird das Installationsdatum in der INI-Datei gespeichert. Später wird bei jedem Start der Applikation in der INI-Datei geprüft, ob das Installationsdatum vorhanden ist. Wenn das Datum fehlt und wenn die Datei „FirstInstall.txt" nicht vorhanden ist, wird davon ausgegangen, dass an der Installation manipuliert wurde und die Ausführung der Applikation wird beendet.

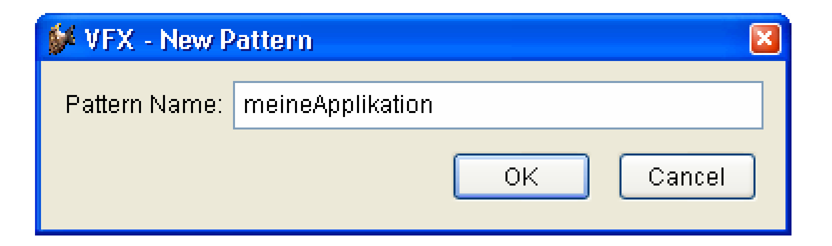
Wenn die Datei „FirstInstall.txt" nicht verwendet wird, wird die Ini-Datei neu erstellt, falls sie nicht vorhanden ist.

Das Installationsdatum kann auf zwei Arten ermittelt werden. Entweder wird das Systemdatum verwendet oder es wird das Erstellungsdatum einer bestimmten Datei verwendet. Wenn das Erstellungsdatum einer Datei verwen-

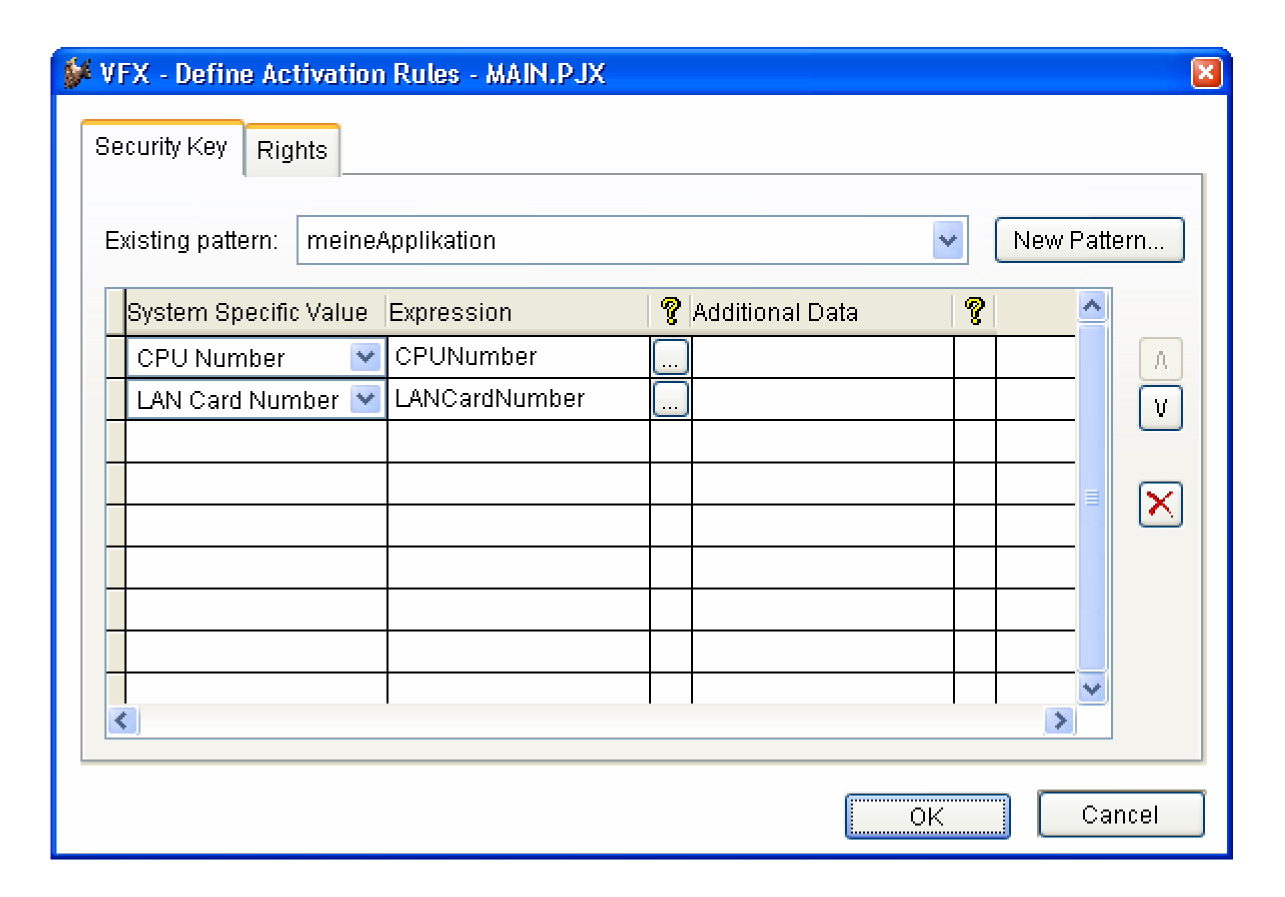
det werden soll, muss der Name dieser Datei in der Eigenschaft *cRegFileName* der Klasse cVFXActivation gespeichert werden.

10.3. Die Definition der Aktivierungsregeln

Wenn der „Define Activation Rules"-Assistent das erste Mal für ein Projekt gestartet wird, muss eine neue Regel für dieses Projekt angelegt werden.



Nach der Eingabe eines Namens für die Regel wird der „Define Activation Rules"-Assistent gestartet.



Auf der Seite Security Key des Assistenten befindet sich eine Combobox aus der eine Regel für das aktuelle Projekt ausgewählt werden kann. In dem darunterliegenden Grid können so viele Zeilen hinzugefügt werden, wie benötigt werden. Aus allen Zeilen des Grids wird in ein Schlüssel generiert, der in

der Eigenschaft cactpattern der Klasse cVfxactivation gespeichert wird. Die Anwendung beim Kunden erkennt anhand dieses Schlüssels welche systemspezifischen Werte des PCs zur Generierung des Installationsschlüssels verwendet werden müssen. Der Installationsschlüssel stellt sicher, dass die Applikation nur auf dem Computer ausgeführt wird, für den der Aktivierungsschlüssel erstellt wurde.

In der ersten Spalte des Grid kann ein systemspezifischer Wert ausgewählt werden. In einer Combobox sind hier alle möglichen Parameter aufgeführt, die zur Erstellung des Installationsschlüssels verwendet werden können. Zusätzlich können Zeichenkettenfunktionen angewendet werden, um den Wert zu verändern.

Zum Beispiel sollen anstelle der vollständigen Seriennummer einer Festplatte nur die letzten vier Stellen zur Erstellung des Installationsschlüssels verwendet werden. Aus der Combobox in der ersten Spalte wird „HDD Factory Serial Number“ ausgewählt. Die VFX-Systemvariable, die diesem Parameter entspricht heißt „HDDFactoryNumber“ und erscheint in der zweiten Spalte. Um nur die letzten vier Stellen zu verwenden, muss der folgende Ausdruck in der zweiten Spalte eingetragen werden: RIGHT(ALLTRIM(HDDFactoryNumber),4).

Wenn einer der systemspezifischen Werte „File Creation Date“ oder „Registry Key Value“ verwendet werden soll, müssen weitere Parameter angegeben werden. Wenn das Erstellungsdatum einer Datei verwendet werden soll, muss der Name der Datei angegeben werden. Um einen Windows-Registrierungsschlüssel verwenden zu können, muss die Bezeichnung des Schlüssels eingegeben werden. Dies geschieht in der Spalte „Additional Data“.

Aus den Aktivierungsregeln wird auf dem PC des Anwenders ein Installationsschlüssel erstellt. Dabei werden alle in den Aktivierungsregeln enthaltenen Parameter berücksichtigt. Wenn nur ein Parameter auf dem PC des Anwenders verändert wird, wird die Installation ungültig und der Anwender muss einen neuen Aktivierungsschlüssel anfordern, entsprechend seiner geänderten Hardware.

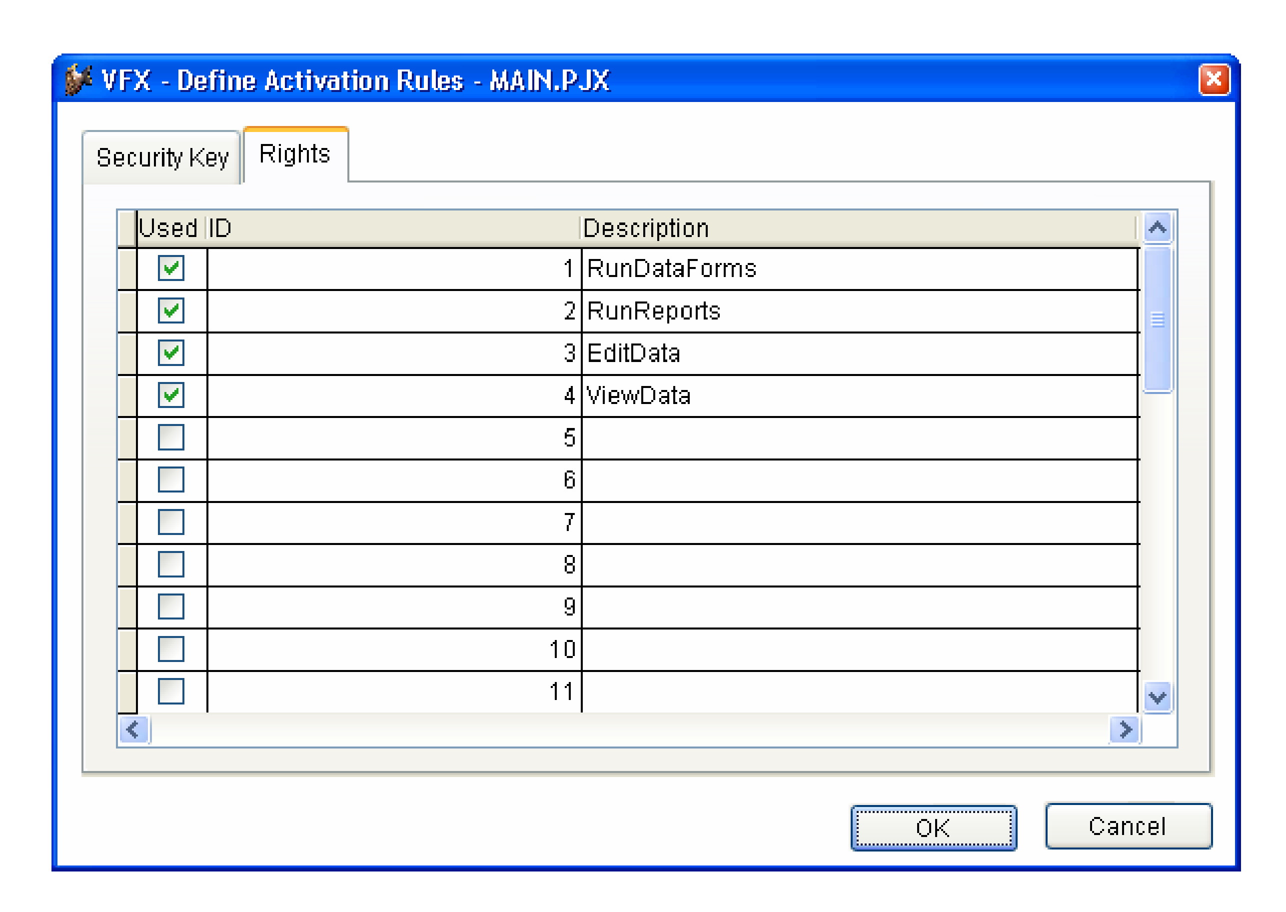
Es können so viele Zeilen dem Grid hinzugefügt werden, wie benötigt werden. Die Zeilen im Grid können mit den Pfeiltasten am rechten Rand des Assistenten in eine andere Reihenfolge gebracht werden. Durch verschieben der Zeilen im Grid ändern sich die Aktivierungsregeln.

HINWEIS: Je mehr Zeilen dem Grid hinzugefügt werden, desto länger werden die Aktivierungsschlüssel!

Nach der Definition der Aktivierungsregeln wird das Muster in der Eigenschaft cActPattern der Klasse cVFXActivation (Appl.vcx) gespeichert.

ACHTUNG: Der Wert der Eigenschaft *cActPattern* darf niemals gelöscht werden! Ohne diesen Wert ist es nicht möglich Aktivierungsschlüssel zu erstellen!

Auf der Seite *Rights* können bis zu 32 verschiedene Benutzerrechte angelegt werden. Damit kann der Zugriff auf bis zu 32 Module einer Applikation gesteuert werden. Beispielsweise können Rechte angelegt werden, die es dem Anwender erlauben Formulare zu starten (*RunDataForms*), Berichte zu drucken (*RunReports*), Daten zu bearbeiten (*EditData*), Daten anzusehen (*ViewData*) usw. Zur Laufzeit der Applikation können die einzelnen Berechtigungen überprüft werden und ggf. wird die entsprechende Aktion ausgeführt.



Alle Benutzerrechte stehen zur Laufzeit als Eigenschaften des global sichtbaren Objekts goProgram.SecurityRights zur Verfügung, sodass an jeder Stelle der Applikation darauf zugegriffen werden kann.

Wenn die Applikation nicht aktiviert ist, haben alle Benutzerrechte den Wert -1. Wenn die Applikation aktiviert ist, hat ein Benutzerrecht den Wert 1, wenn die Aktion erlaubt ist und 0, wenn die Aktion nicht erlaubt ist.

Um im Assistenten ein Recht einzutragen muss zuerst das Kontrollkästchen in der ersten Spalte markiert werden. Dann wird ein Name für das Recht ein-

getragen. Zur Laufzeit der Applikation wird eine Eigenschaft des Security-Rights-Objekts mit diesem Namen angelegt. Daher müssen bei der Eingabe des Namens die Konventionen zur Namensgebung von VFP beachtet werden!

Anmerkung: Applikationsrechte sind für jede Applikation unterschiedlich. Die Rechte, die für eine andere Applikation erstellt wurden, können nicht verwendet werden. Auch wenn ähnliche Rechte benötigt werden, müssen diese neu erstellt werden. Die Applikationsrechte werden in der Tabelle Vfxapprights.dbf im Projektordner gespeichert.

10.4. Erstellen eines Aktivierungsschlüssels

Wenn der Anwender seinen Installationsschlüssel sendet, muss ein Aktivierungsschlüssel erstellt werden. Dieser Aktivierungsschlüssel teilt der Applikation mit, ob der Anwender eine bestimmte Aktion ausführen darf. Für jede Aktion muss das entsprechende Recht ausgewählt werden.

Wenn aus dem VFX 8.0-Menü „Create Activation Key“ aufgerufen wird, erscheint der Dialog mit dem Benutzerrechten für das aktive Projekt.

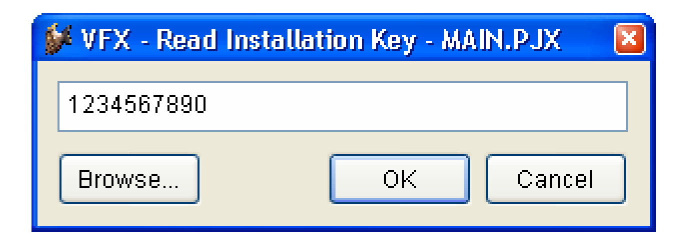
VFX - Create Activation Key - MAIN.PJX

ID	Description	User has this right
1	RunDataForms	☑
2	RunReports	☐
3	EditData	☑
4	ViewData	☑
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		

Read Installation Key OK Cancel

Mit der Schaltfläche „Read Installation Key“ öffnet sich ein Dialog, in den der Installationsschlüssel des Anwenders eingegeben wird. Der Installations-

schlüssel kann über die Zwischenablage eingefügt werden oder aus einer Datei gelesen werden.



Nachdem jedes für den Anwender erlaubte Recht markiert ist, wird mit einem Klick auf OK der Aktivierungsschlüssel generiert. Der erstellte Aktivierungsschlüssel wird in der Datei <Projektname>.xak im Projektordner gespeichert. Der Aktivierungsschlüssel oder die Datei muss an den Anwender zur Aktivierung der Applikation gesendet werden.

Wenn dem Anwender entsprechend dem obigen Beispiel alle Rechte zur Datenbearbeitung gegeben wurden, er aber nicht das Recht hat Berichte zu drucken, sehen die Eigenschaften zur Laufzeit so aus:

```
goProgram.SecurityRights.RunDataForms = 1
goProgram.SecurityRights.RunReports = 0
goProgram.SecurityRights.EditData = 1
goProgram.SecurityRights.ViewData = 1
```

Wenn der Anwender eine Applikation startet, die eine Aktivierung erfordert (und wenn die Applikation noch nicht aktiviert wurde), wird automatisch der Installationsschlüssel erzeugt. Abhängig vom Wert der Eigenschaft *nRegWay* wird der Installationsschlüssel entweder angezeigt oder in einer Datei gespeichert, die per E-Mail versendet werden kann. Nachdem der Anwender den Aktivierungsschlüssel erhalten hat, kann er ihn im Aktivierungsfenster eingeben oder die Datei mit dem Aktivierungsschlüssel im Projektordner speichern. Damit ist die Applikation auf diesem Computer aktiviert.

Wenn der Anwender später den Menüpunkt *Hilfe, Produkt aktivieren* auswählt, wird der Installationsschlüssel angezeigt, unabhängig von der Einstellung der Eigenschaft *nRegWay*.

10.5. Eigenschaften der Klasse cVFXActivation

cFirstInstall – Diese Eigenschaft enthält den Namen einer Datei. Anhand des Vorhandenseins dieser Datei entscheidet diese Klasse, ob die Applikation erstmalig gestartet wird. Wenn dieser Eigenschaft eine leere Zeichenkette zugewiesen wird, kann nicht überprüft werden, ob die Applikation erstmalig gestartet wird. Das Datum des Starts wird dann ohne weitere Überprüfung in der Ini-Datei eingetragen.

cINIFileName – Der Name der Ini-Datei, in der die Aktivierungsinformationen und das Datum des ersten Applikationsstarts gespeichert sind. Der Standardwert ist „VFX.INI".

cParamFile – Der Name der Datei, in der der Installationsschlüssel gespeichert wird. Abhängig vom Wert der Eigenschaft *nRegWay* kann diese Datei per E-Mail versendet oder auf einem anderen Weg verarbeitet werden.

cRegMail – In dieser Eigenschaft wird die E-Mail-Adresse des Entwicklers gespeichert, an die die Datei mit dem Installationsschlüssel gesendet wird, wenn die Eigenschaft *nRegWay* den Wert 2 hat.

cRegFileName – Hier kann der Name einer Datei angegeben werden, die bei der Installation erstellt wird. Das Erstellungsdatum dieser Datei wird verwendet um das Installationsdatum zu ermitteln. Wenn dieser Eigenschaft kein Wert zugewiesen wird, wird das Systemdatum des ersten Starts der Anwendung verwendet.

nRegWay – In dieser Eigenschaft kann eingestellt werden, wie der Entwickler den Installationsschlüssel bekommen soll.

0 – Der Installationsschlüssel wird in einem Dialog angezeigt und der Anwender kann den Installationsschlüssel kopieren und in beliebige Applikationen einfügen.

1 – Der Installationsschlüssel wird in einer Datei gespeichert. Der Anwender kann diese Datei später an den Entwickler übermitteln. Der Name der Datei wird in der Eigenschaft cParamFile hinterlegt.

2 – Der Installationsschlüssel wird in einer Datei gespeichert und an den Entwickler als E-Mail-Anhang gesendet. Der Name der Datei wird in der Eigenschaft cParamFile hinterlegt. Die E-Mail-Adresse des Entwicklers, an die der Installationsschlüssel gesendet wird, wird in der Eigenschaft cRegEMail eingetragen.

11. Weitere Entwicklungstechniken

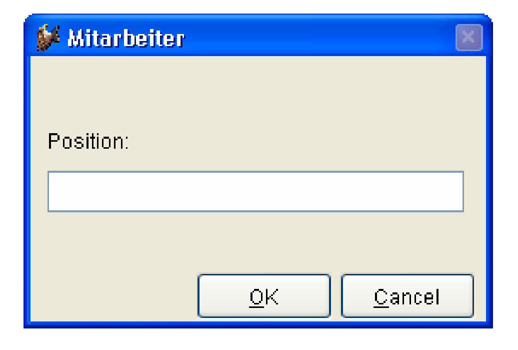
11.1. Formulare basierend auf Ansichten

Bei der Entwicklung von VFX wurde großer Wert darauf gelegt, dass sowohl direkt mit VFP-Tabellen, als auch mit lokalen Ansichten und mit Remote Ansichten gearbeitet werden kann. Wenn die Datenquelle eines Formulars eine Ansicht sein soll, muss auf der Seite Optionen des VFX - Form Builder das Häkchen bei „Work On View" gesetzt werden. Damit weiß VFX, dass es sich bei der Datenquelle um eine Ansicht handelt. Ansichten können insbesondere keine Indexschlüssel haben. VFX muss also in jedem Fall, in dem eine Sortierung benötigt wird, eine temporäre Indexdatei erstellen.

11.1.1. Eingabe der Ansichtsparameter – CAskViewArg

In den meisten Fällen sind Ansichten parametrisiert. Die Parameter müssen vor Abfrage der Daten der Ansicht bekannt sein. Zur Eingabe der Ansichtsparameter stellt VFX die Formularklasse *CAskViewArg* zur Verfügung. Das Datenbearbeitungsformular wird wie gewohnt mit dem VFX - Form Builder erstellt. Die Eigenschaft *lworkonview* wird auf .T. gesetzt. Bei der Ansicht in der Datenumgebung wird die Eigenschaft *nodataonload* auf .T. gesetzt. Das bedeutet, dass die Ansicht beim Laden des Formulars geöffnet wird, ohne dass Datenabgefragt werden.

Jetzt wird ein neues Formular basierend auf der Klasse *CAskViewArg* erstellt. Die Steuerelemente, die als Controlsource Felder enthalten, die auch als Ansichtsparameter verwendet werden, können über die Zwischenablage vom Bearbeitungsformular auf das Formular basierend auf der Klasse *CAskViewArg* kopiert werden. In der Eigenschaft *cviewparameter* ist der Name des Ansichtsparameters einzutragen. Den Steuerelementen können geeignete Bezeichnungen hinzugefügt werden. Das Formular ist damit fertig und kann gespeichert werden.



Aus dem Bearbeitungsformular muss nun noch das Formular basierend auf der Klasse *CAskViewArg* aufgerufen werden. Dies geschieht am Ende des I-nit-Events:

```
do form <Formular zur Eingabe der Ansichtsparameter> with this
```

Es ist auch möglich zur Laufzeit des Formulars das Formular zur Eingabe der Ansichtsparameter erneut aufzurufen. Wenn der Aufruf aus einem Steuerelement, zum Beispiel aus dem Click-Event einer Schaltfläche erfolgt, muss der Aufruf so aussehen:

```
do form <Formular zur Eingabe der Ansichtsparameter> with thisform
```

Mehr ist bei der Arbeit mit Ansichten nicht zu beachten. Alles Weitere erledigt VFX.

11.2. CWizard-Klasse

Die Klasse CWizard ermöglicht die Erstellung von Assistenten. Der Anwender wird Schritt für Schritt durch die Bearbeitung geführt. Ein gutes Beispiel für die Verwendung der Klasse CWizard ist in den VFX-Wizards selbst enthalten. Der VFX – Metadata Wizard basiert auf der Klasse CWizard.

11.3. Delayed Instantiation

Die Ladezeit eines Formulars hängt im Wesentlichen von der Anzahl der Steuerelemente ab, die mit dem Formular geladen werden müssen. Nun sind aber in der Regel nicht alle Steuerelemente eines Formulars sofort sichtbar, wenn ein Formular gestartet wird. Wenn mit einem Seitenrahmen gearbeitet wird, sind zunächst nur die Steuerelemente einer Seite sichtbar. Die Steuerelemente der anderen, zunächst nicht sichtbaren Seiten, brauchten also gar nicht geladen werden. Erst wenn der Benutzer erstmals eine andere Seite aktiviert, müssen die auf dieser Seite befindlichen Steuerelemente nachgeladen werden.

Die Delayed Instantiation wird von VFX mit der sehr praktischen Funktion *addpagedelay()* unterstützt.

Um das Ziel zu erreichen müssen zunächst alle Steuerelemente einer Seite eines Pageframes in einem Container als Klasse gespeichert werden. Dafür markiert man im VFP Formular-Designer alle Steuerelemente der aktuellen Seite und wählt im Menü File den Punkt „Save As Class". Die Klasse sollte in der Klassenbibliothek Appl.vcx gespeichert werden. Diese Klassenbibliothek steht dem Entwickler für eigene Klassen zur Verfügung. Beim Speichern als Klasse ergänzt VFP automatisch einen Container um die ausgewählten Steuerelemente. Der Name der Klasse sollte so gewählt werden, dass der Bezug zu dem Formular und der Seite des Pageframes leicht ersichtlich sind. Die als Klasse gespeicherten Steuerelemente können nun von dem Seitenrahmen gelöscht werden.

Um den Container zur Laufzeit des Formulars nachzuladen wird die Funktion *addpagedelay()* verwendet. Der Aufruf muss in das Activate Event der jeweiligen Seite eingefügt werden und sieht so aus:

```
AddPageDelay(thisform, this, 'x', '<Name der Klasse>')
```

Es empfiehlt sich ein Formular zunächst ohne Delayed Instantiation zu entwickeln und zu testen. Wenn das Formular fast fertig ist, kann es auf Delayed Instantiation umgestellt werden. Zu beachten ist dabei, dass Referenzen auf einzelne Steuerelemente geändert werden müssen. Während vor der Umstellung auf Delayed Instantiation auf eine Textbox zum Beispiel so referenziert werden konnte:

```
Thisform.pgfPageframe.Page1.txtMeinetextbox
```

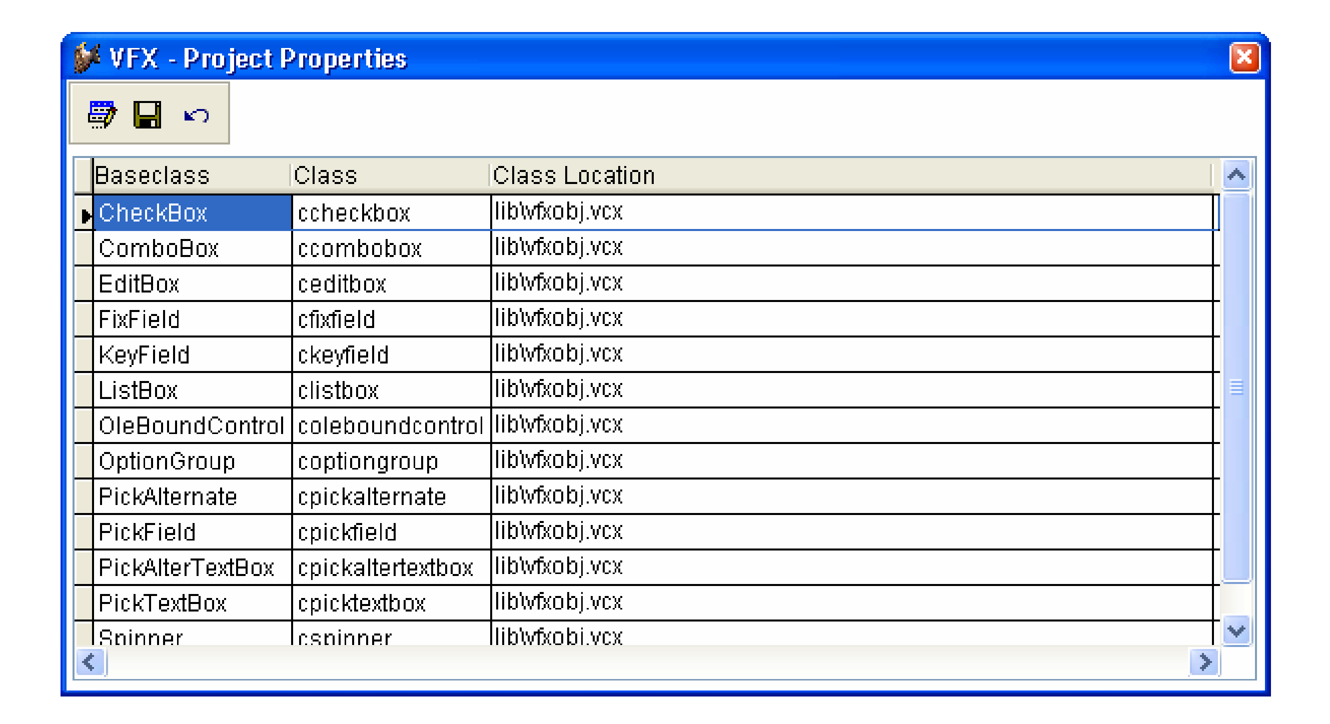
Sieht die Referenz nach Umstellung auf Delayed Instantiation so aus:

```
Thisform.pgfPageframe.Page1.x.txtMeinetextbox
```

Das x ist hierbei der Name des Containers, in dem sich die Steuerelemente der Seite befinden.

11.4. VFX – Project Properties

In VFX können eigene Ableitungen der VFX-Klassen verwendet werden. Im Dialog VFX – Project Properties können die zu verwendenden Klassen für die einzelnen Steuerelement-Typen eingetragen werden. Als Vorgabe stehen hier die bekannten Klassen aus der Klassenbibliothek Vfxobj.vcx. Der VFX-Entwickler kann diese Vorgaben ändern und eigene Klassen, die vorzugsweise in der Klassenbibliothek Appl.vcx gespeichert sind, eintragen. Diese Klassen können von den VFX-Buildern bei der Erstellung neuer Formulare verwendet werden.



Baseclass	Class	Class Location
CheckBox	ccheckbox	lib\vfxobj.vcx
ComboBox	ccombobox	lib\vfxobj.vcx
EditBox	ceditbox	lib\vfxobj.vcx
FixField	cfixfield	lib\vfxobj.vcx
KeyField	ckeyfield	lib\vfxobj.vcx
ListBox	clistbox	lib\vfxobj.vcx
OleBoundControl	coleboundcontrol	lib\vfxobj.vcx
OptionGroup	coptiongroup	lib\vfxobj.vcx
PickAlternate	cpickalternate	lib\vfxobj.vcx
PickField	cpickfield	lib\vfxobj.vcx
PickAlterTextBox	cpickaltertextbox	lib\vfxobj.vcx
PickTextBox	cpicktextbox	lib\vfxobj.vcx
Spinner	cspinner	lib\vfxobj.vcx

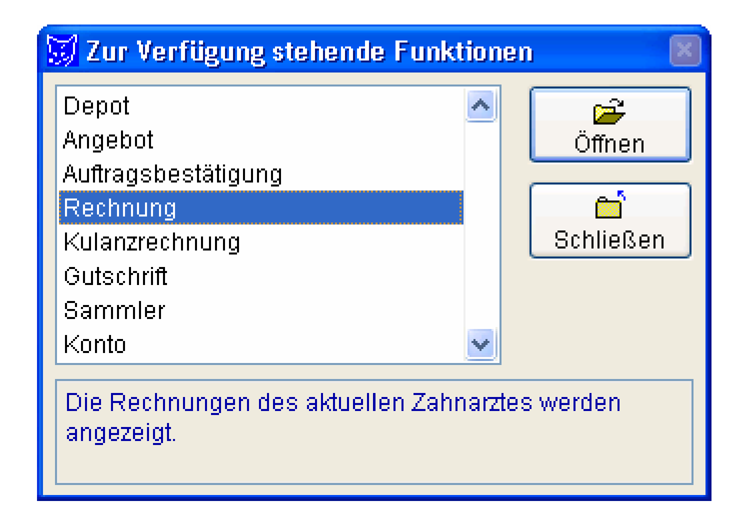
11.5. Wichtige VFX-Methoden

11.5.1. Valid

VFX bietet eine Valid-Methode auf Formularebene. Diese Methode wird immer aufgerufen, wenn die Daten des Formulars gespeichert werden sollen. Hier sollten also alle Validierungen untergebracht werden. Wenn aus dieser Methode der Wert .F. zurückgegeben wird, wird der Speichervorgang nicht fortgesetzt und das Formular bleibt im Bearbeitungsmodus. Durch Rückgabe von .T. werden die Daten gespeichert.

11.5.2. Onmore

Mithilfe dieser Methode ist es insbesondere möglich Child-Formulare aufzurufen. Ein fertiger Template-Code kann auf Wunsch vom VFX – Form Builder im Formular eingetragen werden. Je nach Anwendungsfall brauchen nur noch wenige Werte dieser Methode vom Entwickler angepasst werden.



Über die Onmore-Methode wird zur Laufzeit ein Dialog angezeigt, in dem der Benutzer das aufzurufende Child-Formular auswählen kann.

11.5.3. Onpostinsert

Diese Methode wird unmittelbar nach dem Anfügen eines neuen Datensatzes aufgerufen, noch bevor der Benutzer die Möglichkeit zur Bearbeitung der Daten erhält.

Hier können also Standardvorgaben in den Feldern eingetragen werden. Diese Methode bietet sich auch an, um Primärschlüssel zu vergeben.

11.5.4. Onrecordmove

Jedes Mal, wenn der Satzzeiger bewegt wird, wird diese Methode aufgerufen. Hier können Werte angezeigt oder aktualisiert werden, die nicht aus der Datenbank stammen.

11.6. VFX Primärschlüssel-Generierung

Es kann Tabellen geben, aus denen Sie den Primärschlüssel nicht den Benutzern zeigen wollen. Aber für ein korrektes Datenbankdesign wollen Sie einen Primärschlüssel verwenden. Für diese und ähnliche Situationen bietet VFX eine Funktion, die die Erstellung von Primärschlüsseln ermöglicht und in einer Mehrbenutzerumgebung genauso funktioniert, wie in einer Client/Server-Umgebung.

Durch das modulare Design der VFX-Klassenhierarchie, haben Sie die Möglichkeit, nach dem Einfügen eines neuen Datensatzes einzugreifen. VFX bietet, neben vielen anderen Funktionen, eine Methode mit dem Namen *OnPostInsert(),* die in dem Moment ausgeführt wird, wenn ein neuer Datensatz gerade hinzugefügt wurde. Normalerweise bietet VFX für alle wichtigen Ereignisse Methoden, die automatisch vor, während und nach dem Ereignis ausgeführt werden. In diesem Fall, in dem ein neuer Datensatz hinzugefügt wird, gibt es die folgenden Methoden:

- OnPreInsert()
- OnInsert()
- OnPostInsert()

Außerdem gibt es eine Eigenschaft die angibt, ob der Benutzer einen neuen Datensatz aufnehmen kann. Diese Eigenschaft trägt den Namen *lCanInsert.*

ANMERKUNG: Für weitere Informationen lesen Sie bitte die VFX Technische Referenz.

Um einen Primärschlüssel zu erzeugen, könnten Sie in die *OnPostInsert()*-Methode Ihres Formulars etwa folgenden Code einfügen. Hierdurch wird die Funktion *GetNewId()* aufgerufen. Der Parameter gibt die Tabelle an, für die der Schlüssel generiert wird.

```
DODEFAULT()
REPLACE comp_id WITH GetNewId('CUSTOMER') IN customer
```

Der Zähler für den generierten Schlüssel wird in der Tabelle *VFXSYSID* gespeichert.

11.7. Hinzufügen eines Formulars zum Öffnen-Dialog

VFX bietet einen Vorschlag für einen Öffnen-Dialog an. Selbstverständlich können Sie diesen Dialog an Ihre Bedürfnisse anpassen oder einen eigenen Dialog erstellen.

Zusätzlich zu dem in bisherigen VFX-Versionen vorhandenem Öffnen-Dialog (*Vfxfopen.scx*) steht in VFX 8.0 ein neuer Öffnen-Dialog im Windows-XP-Stil (*Vfxxpopen.scx*) zur Verfügung. Dieser neue Öffnen-Dialog ist standardmäßig aktiviert. Mit der Eigenschaft *goprogram.lxpopenstyle* kann auf Wunsch auf den alten Öffnen-Dialog umgeschaltet werden.



lxpopenstyle

.T. – der neue Öffnen-Dialog im Windows-XP-Stil wird verwendet.
.F. – der alte Öffnen-Dialog (Vfxfopen.scx) wird verwendet.

Die Gruppenüberschriften im neuen Öffnen-Dialog werden aus dem neuen Tabellenfeld *Vfxopen.groupcap* gelesen. Der Zustand der einzelnen Gruppen (aufgeklappt oder zugeklappt) wird je Benutzer gespeichert.

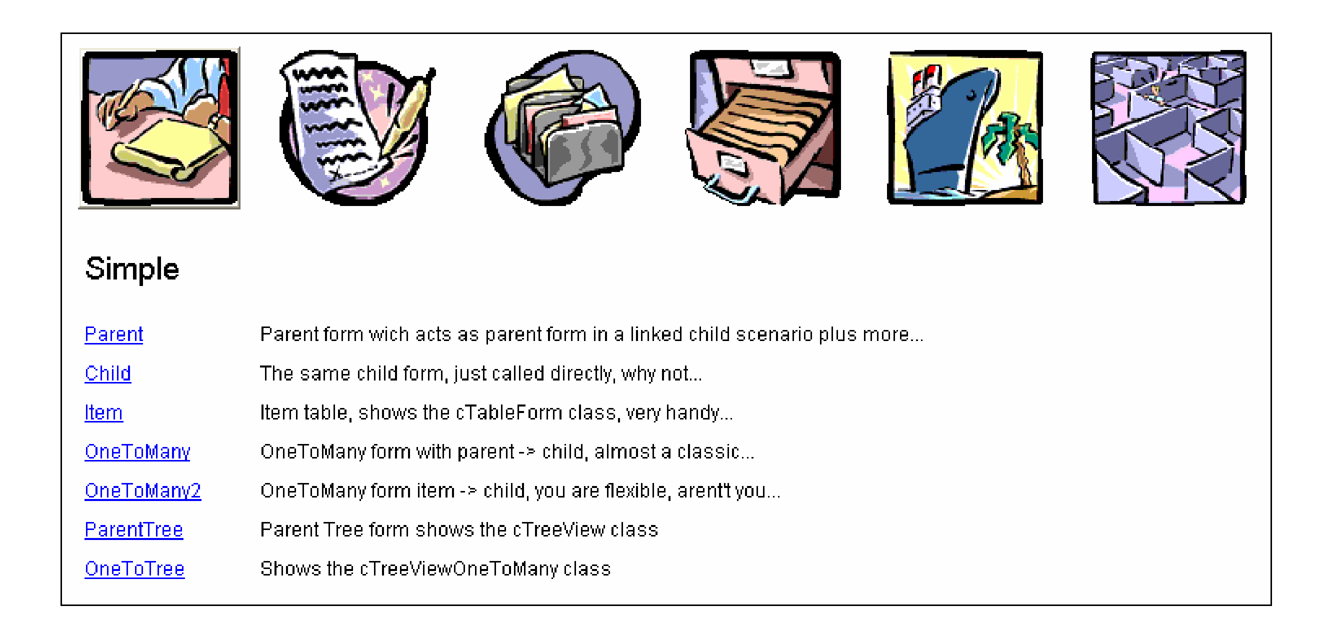
Der Datei/Öffnen-Dialog benutzt die Tabelle *VFXFOPEN.DBF*. Die VFX-Formular-Builder fügen automatisch für jedes Formular einen Datensatz zu der Tabelle *VFXFOPEN.DBF* hinzu. Hier ist die Struktur der Tabelle *VFXFOPEN.DBF*:

VFXFOpen-Feld	**Beschreibung**	**Beispiel**
ObjectID	Diese Feld wird verwendet, wenn der Öffnen-Dialog *Vfxfopen.scx* verwendet wird. Hierzu muss die Eigenschaft *goprogram .lxpopenstyle*=.*F*. gesetzt sein. Der VFX-Öffnen-Dialog hat normalerweise zwei Seiten. (**Tipp:** Sie können die *Pagecount*-Eigenschaft des Seitenrahmens im Formular *Vfxopen.scx* auf jeden beliebigen Wert setzen, um die Anzahl der Seiten zu verändern.) Wenn Sie wollen, dass Ihr Formular auf Seite 1 des Seitenrahmens erscheint, geben Sie PAGE1 ein. Für die weiteren Seiten PAGE2, PAGE3 usw.	PAGE1
ObjectNo	Geben Sie eine Zahl für die Sortierfolge der Liste ein. 1 wird das erste Element, es folgt 2 usw. Die Sortierung wird auf jeder Seite benutzt.	1
GroupCap	Dieses Feld wird verwendet, wenn der Öffnen-Dialog Vfxxpopen.scx verwendet wird. Hierzu muss die Eigenschaft goprogram .lxpopenstyle=.T. gesetzt sein. Dieses Feld enthält eine Gruppenüberschrift. Die Gruppierung erfolgt entsprechend der Einträge im Feld ObjectID. Die GroupCap muss nur für den ersten Eintrag einer Gruppe eingetragen werden.	Kontakte
Title	Geben Sie die Überschrift ein, die im Listenfenster erscheint.	Kunden
Descr	Geben Sie einen Beschreibungstext ein, der angezeigt wird, wenn der Benutzer diesen Eintrag ausgewählt hat.	Liste aller Adressen
Form	Geben Sie den Namen des aufzurufenden Formulars ein.	ADRE
Parameter	Wenn Sie an das Formular Parameter übergeben wollen, können Sie diese hier eingeben.	
Viewlevel	Die Benutzerstufe, die erforderlich ist, um ein Formular anzusehen (Zum Beispiel 1 = Admin, 2 = Hauptbenutzer, 3 = normaler Benutzer usw.)	1 (nur Administratoren können dieses Formular ansehen)
NewLevel	Die Benutzerstufe, die erforderlich ist, um neue Datensätze dem Formular hinzufügen zu können.	1 (nur Administratoren können neue Datensätze hinzufügen)
EditLevel	Die Benutzerstufe, die erforderlich ist, um Datensätze bearbeiten zu können.	1 (nur Administratoren können Datensätze bearbeiten)
Eraselevel	Die Benutzerstufe, die erforderlich ist um auf diesem Formular Datensätze löschen zu können.	1 (nur Administratoren können Datensätze löschen)

11.8. Active Desktop

Der Active Desktop gibt den Anwendungen ein professionelles Startbild. Auf dem sonst leeren Bildschirm werden Bilder und Auswahlmöglichkeiten angeboten. Durch das Bewegen der Maus über die Bilder wird das zugehörige Menü unterhalb der Bilder angezeigt. In den Menüs befinden sich unterstrichene Menüpunkte, die ähnlich Hyperlinks im Internet Explorer, einfach angeklickt werden können und eine Aktion ausführen. In den meisten Fällen wird als Aktion ein Formular gestartet werden.

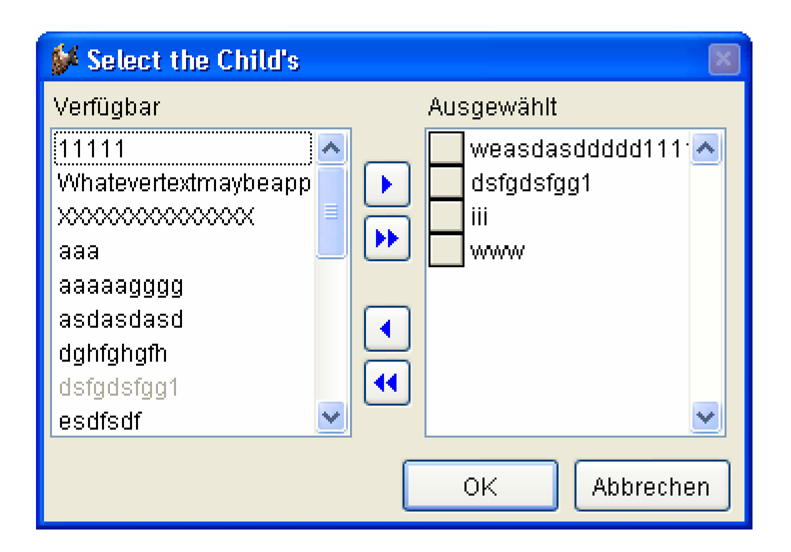
Die Klasse des Active Desktop befindet sich in der Klassenbibliothek *Appl.vcx* und kann nach den Wünschen des Entwicklers um beliebige Steuerelemente erweitert werden.



Der Active Desktop kann zusätzlich oder anstelle des Öffnen-Dialogs verwendet werden.

11.9. Benutzung des VFX-Moverdialogs

Der VFX-Moverdialog ist ein leistungsfähiges Bedienungselement, das Sie in Ihren Anwendungen benutzen können. Der VFX-Moverdialog bekommt als Parameter zwei Arrays übergeben. Das erste Array enthält zur Auswahl stehende Elemente. Diese Elemente werden in der linken Listbox angezeigt. Das zweite Array enthält die ausgewählten Elemente. Das zweite Array kann bei Aufruf des Moverdialogs leer sein. Der Anwender kann eine beliebige Anzahl von Elementen auswählen.



Hier ein Beispielcode für die praktische Anwendung des VFX-Moverdialog-Steuerelements:

```
LOCAL laSource[1,1], loMover

*--prepare the array of all available items
SELECT keygrp_id, keygrp_name FROM keygrp INTO ARRAY laSource

*--create the mover object based on the VFX Class CMoverDialog
loMover = CREATEOBJECT("CMoverDialog")
*--set the caption
loMover.Caption = CAP_KEYFIELDGEN
*--set the property which defines which column from the array get's
displayed
loMover.cntMover.nColToView = 2
*--enable multiple selections
loMover.cntMover.lstSource.MultiSelect = .T.
*--pass the array of all available items
*  here you can also pass a second parameter if you want to define,
which
*  elements from the array must appear as already selected
loMover.cntMover.SetData(@laSource)
*--show the mover dialog
loMover.Show()

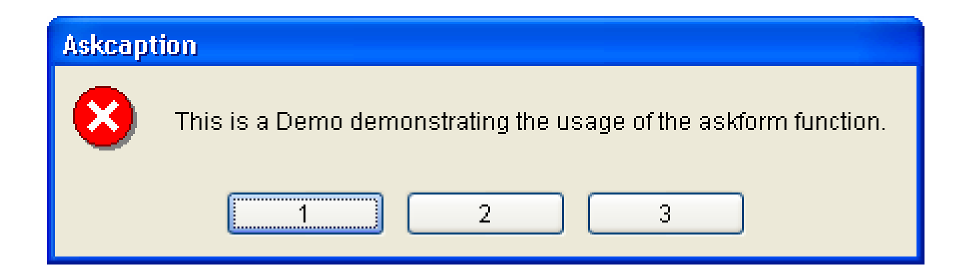
*--Result: The Public Array _gaMoverList contains the selected
items, use it
*  and release this Public Array after you have done.
```

Nach der Erstellung des Objektes *loMover* haben Sie die vollständige Kontrolle darüber und können alle gewünschten Eigenschaften und Methoden verändern.

ANMERKUNG: Um eine detaillierte technische Beschreibung der VFX-Klassenbibliotheken inklusive aller Eigenschaften und Methoden zu erhalten, lesen Sie bitte in der VFX Technischen Referenz nach.

11.10. Askform

Die Askform entspricht in etwa einer Messagebox, hat jedoch eine erweiterte Funktionalität. Die Beschriftungen der (maximal) drei Schaltflächen können als Parameter übergeben werden. Außerdem ist es möglich ein Timeout für die Messagebox festzulegen. Bei erreichen des Timeouts ohne Benutzeraktion wird ein Rückgabewert geliefert, der dem Drücken der Standard-Schaltfläche entspricht.



Ein Beispiel zur Verwendung der Funktion *Askform()* befindet sich im Formular *Parent.scx* aus der Demoapplikation VFX80Test.

11.11. IDX Know How

VFX macht von vorhandenen Indexschlüsseln bestmöglichen Gebrauch. Für die inkrementelle in VFX-Power Grids durchsucht VFX automatisch alle vorhandenen Indexschlüssel der verwendeten Tabelle. Für Zeichenfelder wird ein Indexschlüssel mit *UPPER()*-Klausel erwartet. Für Datumsfelder wird ein Indexschlüssel mit *DTOS()*-Klausel erwartet.

Wenn VFX keinen passenden Indexschlüssel findet, wird eine temporäre Indexdatei angelegt. Diese Indexdatei wird gelöscht, sobald das Formular geschlossen wird. Ferner wird die Indexdatei gelöscht, wenn das Formular in den Bearbeitungsmodus oder in den Einfügemodus wechselt sowie beim Löschen von Datensätzen. Das ist sinnvoll weil laufende Transaktionen, wie sie zum Beispiel im RI-Code verwendet werden, zu VFP-Laufzeitfehlern führen würden, wenn temporäre Indexdateien geöffnet sind. VFP erlaubt keine temporären Indexdateien, wenn mit Transaktionen gearbeitet wird.

Wenn in einem Formular Transaktionen verwendet werden, kann auf Wunsch nach der Datenbearbeitung der zuvor gültige Indexschlüssel wieder erstellt werden. Dem Anwender wird vorgetäuscht, dass die gewählte Sortierfolge ständig erhalten bleibt. Stellen Sie dafür in Vfxmain.prg ein:

```
lremakeidxafterclear = .T. && Index nach der Bearbeitung wieder
erstellen.
```

Wenn in einem Formular und jeglichem daraus aufgerufenen Code keine Transaktionen ausgeführt werden, also in den beteiligten Tabellen auch kein RI-Code hinterlegt ist, können Sie in Vfxmain.prg einstellen, dass temporäre Indexdateien bei der Datenbearbeitung nicht gelöscht werden:

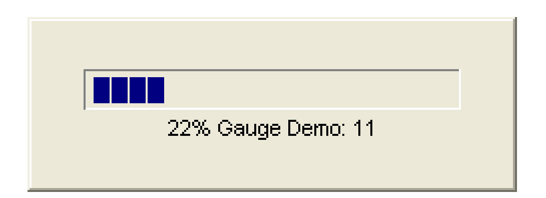
```
lnoclearidxonedit = .t.  && Index zum bearbeiten nicht löschen.
lnoclearidxoninsert = .t.  && Index zum einfügen nicht löschen.
lnoclearidxondelete = .t.  && Index zum löschen von Datensätzen
nicht löschen.
```

Temporäre Indexdateien werden in jedem Fall beim Schließen eines Formulars gelöscht.

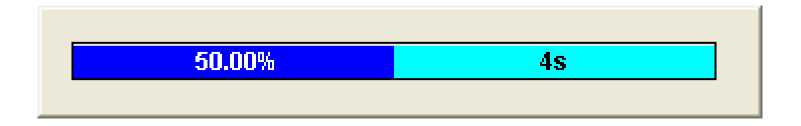
11.12. Fortschrittsanzeige

VFX bietet 2 Möglichkeiten den Fortschritt von lange andauernden Vorgängen zu verdeutlichen.

Die einfache Variante, realisiert mit der Formularklasse *cGaugeWin*, zeigt einen Balken zur Anzeige des Fortschritts an.



Mit dem Formular *Vfxmtr.scx* kann eine Fortschrittsanzeige mit Anzeige der Restzeit dargestellt werden.

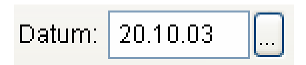


Beispiele für die Verwendung beider Fortschrittsanzeigen befinden sich im Formular *Parent.scx* der Demoapplikation VFX80Test.

11.13. Datumsauswahl

11.13.1. Die Klasse cPickDate

Dis Klasse *cPickDate* enthält eine Textbox zur Eingabe eines Datus sowie eine Schaltfläche zum Aufruf eines Kalenders.



In der Textbox stehen die folgenden Hotkeys zur Auswahl eines Datums zur Verfügung:

+	Nächster Tag
-	Vorheriger Tag
H, h	Heute
B, b	Der erste Tag (Beginn) des angezeigten Monats
L, l	Der letzte Tag des angezeigten Monats
A, a	Neujahr
E, e	Sylvester
V, v	Vorheriger Monat
N, n	Nächster Monat

Für den Kalender wird das ActiveX-Control Microsoft MonthView verwendet. Bei der Erstellung eines Setups muss dieses ActiveX-Control (Mscomct2.ocx) mit in das Setup einbezogen werden. VFP 8 stellt hierfür ein Merge Module bereit.



11.13.2. Die Klasse cDatetime

Zusätzlich steht die Klasse cDatetime zur Eingabe von Datetime-Werten zur Vergügung.



In dieser Klasse ist zur Eingabe des Datums ein cPickDate-Steuerelement enthalten. Es stehen alle Funktionen des cPickDate-Steuerlements wie zum Beispiel der Kalender oder die Hotkeys zur Verfügung.

Um eine Zeiteingabe im 24-Stunden-Format zu ermöglichen muss SET HOURS TO 24 eingestellt sein. Diese Einstellung kann global für alle Formulare in der Funktion *formsetup()* in Applfunc.prg gemacht werden.

Die Controlsource der Klasse cDatetime wird in der Eigenschaft ccontrolsource eingestellt. Die Controlsource muss vom Typ Datetime sein.

11.14. Auswahl von Berichten

Wenn zu einem Formular verschiedene Berichte gedruckt werden sollen, bietet die Klasse *cRSelection* einen geeigneten Auswahldialog. Die zur Verfügung stehenden Berichte werden aus Tabellen gelesen. Es kann zwischen Berichten unterschieden werden, die für alle Benutzer sichtbar sind und Berichten, die nur für einzelne Benutzer sichtbar sind.

Ein Beispiel zur Anwendung findet sich im Formular *Reports.scx* in der Demoapplikation VFX80Test.

11.15. Die Microsoft Agents

Die Agents sind nette Charaktere, die die Benutzung von VFX-Anwendungen auflockern.

In VFX80Test zeigt das Formular *Agent.scx* einfache Beispiele für die Verwendungsmöglichkeiten.

11.16. Linked Child-Formulare

Eine besondere Stärke von VFX ist die Verwendung der Linked Child-Technik. Dabei werden zwei Formulare logisch miteinander verbunden. Ein Formular dient dabei als Parent-Formular. Als Parent-Formular kann jede VFX-Formularklasse dienen. Auch das Child-Formular kann auf jeder VFX-Formularklasse basieren.

Beim Bewegen des Satzzeigers im Parent-Formular wird die Ansicht im Child-Formular automatisch aktualisiert und es werden die zum aktuellen Parent gehörenden Datensätze angezeigt.

Wenn das Child-Formular auf einer Tabelle basiert, wird ein Filter verwendet, um den sichtbaren Datenbereich einzuschränken. Wenn das Child-Formular auf einer Ansicht basiert, wird bei Bedarf ein REQUERY() durchgeführt um die gewünschte Datenmenge anzuzeigen. Die zugrunde liegende Ansicht darf dabei genau einen variablen Ansichtsparameter haben, der dem Parent-Schlüssel entsprechen muss.

Ein Parent-Formular kann mehrere, verschiedene Child-Formulare aufrufen. Ein Child-Formular kann wiederum als Parent für andere Child-Formulare dienen.

Der VFX-Entwickler muss dazu im Child-Formular mit dem Form Builder auf der Seite Optionen „Is Child Form" auswählen oder manuell die Formulareigenschaft *lchildform* auf .T. zu setzen.

Beim Parent-Formular müssen mit dem Form Builder die Optionen „Has More Options" (setzt die Eigenschaft *lmore* auf .T.), „Has Child Form" und „Auto Sync Child Form" (setzt die Eigenschaft *lautosynchildform* auf .T.) ausgewählt werden. Der Form Builder trägt automatisch Template-Code in die Methoden *onmore* und *onsetchilddata* ein. Der Code dieser Methoden muss anschließend manuell bearbeitet werden. In der Methode *onmore* wird das Child-Formular aufgerufen.

11.16.1. Erstellen eines Formulars, das ein Child-Formular aufruft

Obwohl es einen speziellen VFX-Builder zur Erstellung von 1:n-Formularen gibt, ist es manchmal besser, Child-Daten in einem eigenen Formular zu bearbeiten. Das ist insbesondere dann der Fall, wenn Sie das Child-Formular

auch für die direkte Bearbeitung einsetzen und nicht nur durch das Hauptformular einsetzen wollen. Wenn Sie außerdem viele Felder auf dem Child-Formular haben, kann es schwierig werden, diese in einem 1:n-Formular zu bearbeiten.

Im Abschnitt über den VFX-Formular-Builder haben wir bereits das Kontrollkästchen mit dem Namen *Has More Functions* betrachtet. Wenn Sie dieses Kontrollkästchen markieren, generiert der VFX-Formular-Builder den folgenden Code in der *OnMore()*-Methode des Formulars:

```
lparameters tnPassThrough

local lcCalledBy, lcFixFieldValue, lcCaption,;
       lcFixFieldName, lcFilterExpr

lcCalledBy      = ""
lcFixFieldValue = ""
lcCaption       = ""
lcFixFieldName  = ""
lcFilterExpr    = ""

local laFunct[1,5]

laFunct[1,1] = "<Function Title>"
laFunct[1,2] = "<Function Desctiption>"
laFunct[1,3] = "W"    && W - Wait Window, F - Form to run, M -
Method of this form
laFunct[1,4] = "<FormName>"
laFunct[1,5] = lcCalledBy      + ";" +;
               lcFixFieldValue + ";" +;
               lcCaption       + ";" +;
               lcFixFieldName  + ";" +;
               lcFilterExpr

if alen(laFunct,1) = 1
     tnPassThrough = 1
endif

if empty(tnPassThrough)
     do form VFXMORE with laFunct, this
else
     do form VFXMORE with laFunct, tnPassThrough,this
endif

goProgram.RefreshWindowMenu()
```

Dieser Vorlagencode kann so aussehen, wenn Sie ihn an Ihre Bedürfnisse angepasst haben:

```
lparameters tnPassThrough

local lcCalledBy, lcFixFieldValue, lcCaption,;
       lcFixFieldName, lcFilterExpr

lcCalledBy      = "PARENT"
lcFixFieldValue = PARENTID
lcCaption       = "Child records for " + trim(parent.descr)
lcFixFieldName  = "PARENTID"
lcFilterExpr    = "PARENTID='"+parentid+"'"

local laFunct[1,5]

laFunct[1,1] = "Child Records"
laFunct[1,2] = "Child Records for selected parent"
laFunct[1,3] = "F"         && W - Wait Window, F - Form to run, M -
Method of this form
laFunct[1,4] = "CHILD"
laFunct[1,5] = lcCalledBy      + ";" +;
                   lcFixFieldValue + ";" +;
                   lcCaption       + ";" +;
                   lcFixFieldName  + ";" +;
                   lcFilterExpr

if alen(laFunct,1) = 1
     tnPassThrough = 1
endif

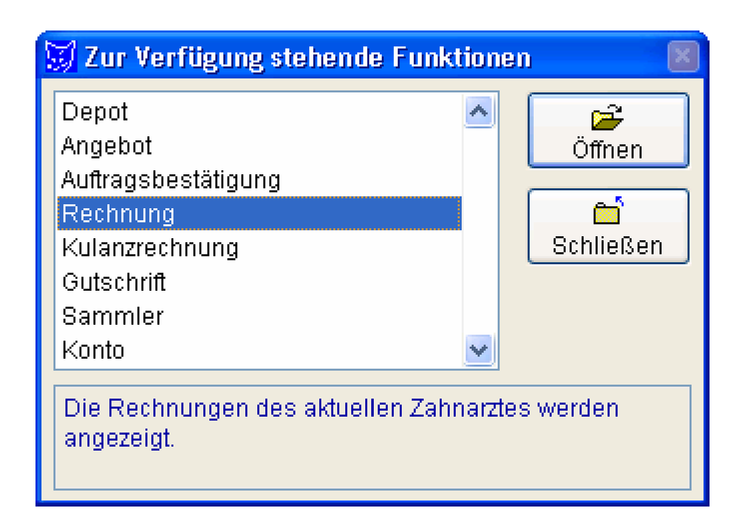
if empty(tnPassThrough)
     do form VFXMORE with laFunct, this
else
     do form VFXMORE with laFunct, tnPassThrough,this
endif

goProgram.RefreshWindowMenu()
```

Wenn der Benutzer die verfügbaren Optionen zum aktuellen Datensatz sehen will, hat er verschiedene Möglichkeiten:

- Er kann die Funktionstaste F6 drücken.
- Er wählt Weitere Funktionen... im Bearbeiten-Menü.
- Er drückt auf die Weitere Funktionen-Schaltfläche in der Standard-Symbolleiste.

Abhängig von dem Code in der Methode *OnMore()* wird der Benutzer einen Dialog sehen, der so ähnlich aussieht wie der folgende:



Der Aufruf der *OnMore()*-Methode mit dem Parameter *tnPassThrough* ist sehr nützlich, wenn Sie ein Formular direkt über die zugeordnete Zahl starten wollen. Sie können diese Technik benutzen, um ein Formular aus der *OnMore()*-Methode über eine Schaltfläche aus einer Symbolleiste zu starten.

Wenn es nur eine Option in der *OnMore()*-Methode gibt, wird das zugeordnete Formular geöffnet, ohne dass dieser Dialog erscheint.

11.16.2. Erstellen eines Child-Formulars

Das Gegenstück eines Formulars, das ein anderes Formular aufruft, ist das aufgerufene Formular. Wie in einem vorangegangenen Kapitel beschrieben, kann es verschiedene Gründe geben, aus denen ein Formular von einem anderen Formular aufgerufen wird.

Wenn Sie ein Formular aufrufen, übergeben Sie die benötigten Parameter an die *Init()*-Methode dieses Formulars. Da die übergebenen Parameter nicht automatisch für andere Methoden des gleichen Formulars sichtbar sind, speichern VFX-Formulare die benötigten Parameter in speziellen Eigenschaften.

Hier ist der Code der *Init()*-Methode, den der VFX-Formular-Builder als Vorlage für Ihre Bedürfnisse erzeugt:

```
lparameters tcArg

local lInitOk

if !empty(tcArg)
      if getArgCount(tcArg) <> 0

            this.cCalledBy      = upper(  getArg(tcArg,1)        )
            this.cFixFieldValue = strtran(getArg(tcArg,2),"@",";")
            this.Caption        =         getArg(tcArg,3)
            this.cFixFieldName  = strtran(getArg(tcArg,4),"@",";")
            this.cFilterExpr    = upper(  getArg(tcArg,5)        )

            this.lPutInLastFile  = .f.


      **************************************************************
            ** Set who has called you

            if this.cCalledBy = "<CalledBy>"

      **************************************************************
            ** Disable CPickField that are Fix Fields for this form

                  *{PickFieldList}*
            endif
      endif
endif

this.SetQueryArg()

lInitOk =eval(this.class+"::init(tcArg)")

********************************************************
** Insert your extra initialization code here

return lInitOk
```

Der Vorlagencode kann so aussehen, wenn Sie ihn an Ihre Bedürfnisse angepasst haben:

```
lparameters tcArg

local lInitOk

if !empty(tcArg)

      if getArgCount(tcArg) <> 0
            this.cCalledBy      = upper(  getArg(tcArg,1)        )
            this.cFixFieldValue = strtran(getArg(tcArg,2),"@",";")
            this.Caption        =         getArg(tcArg,3)
            this.cFixFieldName  = strtran(getArg(tcArg,4),"@",";")
            this.cFilterExpr    = upper(  getArg(tcArg,5)        )

            this.lPutInLastFile  = .f.

      ***************************************************************
            ** Set who has called you

            if this.cCalledBy = "PARENT"

      ***************************************************************
            ** Disable CPickField that are Fix Fields for this form

            ThisForm.pgfPageFrame.Page1.cntParentid.lFixField = .t.

            endif
      endif
endif

this.SetQueryArg()

lInitOk =eval(this.class+"::init(tcArg)")

********************************************************
** Insert your extra initialization code here

return lInitOk
```

Die VFX-Funktion *getArg()* überprüft die Parameterzeichenkette und zerlegt sie in ihre Bestandteile. Die Bestandteile sind durch Semikolon getrennt.

Sehen Sie sich das Beispiel an. Der übergebene Parameter kann die folgende Zusammensetzung haben, wenn wir das Kontakt-Formular zu einer bestimmten Firma aufrufen:

```
"COMP;1234568890;Kontakte zur Firma
DEAG;CONT_COMP_ID;UPPER(CONT_COMP_ID)= '1234568890'"
```

Die individuellen Teile dieser Zeichenkette werden in den bereitgestellten Formulareigenschaften gespeichert, bevor sie innerhalb des ganzen Formulars benutzt werden können. Lassen Sie uns die Formulareigenschaften anschauen, die die Informationen aus der übergebenen Parameterzeichenkette *tcArg* speichern:

VFX-Formular-eigenschaft	Beschreibung	Beispiel
cCalledBy	Diese Zeichenkette gibt an, aus welchem Formular dieses Formular aufgerufen wurde.	COMP
cFixFieldValue	Der Wert des Feldes aus der Haupttabelle (**Primärschlüssel in der Haupttabelle**).	1234568890
cFixFieldName	Der Name des Feldes in der Child-Tabelle, der die 1:n-Beziehung definiert. Dieses Feld erhält den oben angegebenen Wert, wenn ein neuer Datensatz hinzugefügt wird (**Fremdschlüssel in der Child-Tabelle**).	CONT_COMP_ID
cFilterExpr	Der (idealerweise) Rushmore-optimierte Filterausdruck, um die Datensätze entsprechend dem Kriterium der Haupttabelle anzuzeigen.	UPPER(CONT_COMP_ID)= '1234568890'

Bei sehr großen Datenmengen kann es besser sein, mit Ansichten zu arbeiten. Die VFX-Mechanismen arbeiten grundsätzlich genauso. Wenn die Child-Daten aus einer Ansicht stammen, brauchen Sie den Filterausdruck nicht zu übergeben.

11.17. Die VFX-Ressourcentabelle

VFX-Anwendungen verwenden eine Ressourcentabelle, in der je Benutzer Informationen über alle Formulare, die der Benutzer bereits einmal verwendet hat, gespeichert sind. Hierbei werden nicht nur die Positionen der Formulare, sondern auch Layoutänderungen an Grids inklusive der Sortierfolgen gespeichert.

Hier die Einstellungen, die in der VFX-Ressourcentabelle je Benutzer gespeichert werden.

Einstellung	Beschreibung	Bemerkung
Position und Größe von Formularen	Der Benutzer sieht die Formulare bei erneutem Öffnen genau so, wie er sie zuletzt verlassen hat.	Individuelle Formulareinstellungen **Hinweis:** Bezieht sich auch auf Auswahllisten!
Alle vorgenommenen Layoutänderungen an Grids	Der Benutzer sieht die Grids genau so, wie er sie verlassen hat. Sowohl Spaltenbreiten als auch Anordnung (auch wenn es sich hierbei um berechnete Felder handelt).	Individuelle Grid-Einstellungen **Hinweis:** Bezieht sich auch auf Auswahllisten sowie 1:n-Formulare mit mehreren Child-Grid!
Aktuelle Sortierung der Datenbearbeitungsformulare sowie der Auswahllisten	Die letzte Sortierfolge wird automatisch wiederhergestellt. Unabhängig davon, ob ein Indexschlüssel vorhanden ist oder nicht. VFX erstellt temporäre IDX-Dateien für nicht vorhandene Schlüssel.	VFX erstellt automatisch benötigte IDX-Dateien im Ordner der Anwendung und löscht diese wieder beim Verlassen des Formulars. **Hinweis:** Bezieht sich auch auf Auswahllisten!
Position und Status von Symbolleisten	Falls Sie eine Symbolleiste an ein Formular anbinden, so wird diese in demselben Status präsentiert, wie sie beim letzten Arbeiten mit diesem Formular verlassen wurden.	
Unterdrückung von Symbolleisten	Falls der Benutzer die formularspezifische Symbolleiste geschlossen hat, so wird diese bei erneutem Öffnen dieses Formulars nicht mehr geöffnet. Um die Symbolleiste erneut zu aktivieren, muss der Symbolleisten-Dialog aus dem Menü *Ansicht* geöffnet werden und die entsprechende Symbolleiste geöffnet werden.	

Sie können Ihre Ressourcendaten in der Benutzerverwaltung löschen.

VFX-Anwendungen verwenden nicht die Visual FoxPro Ressourcentabelle *FOXUSER.DBF*, stattdessen verwenden Sie ausschließlich die freie VFX-Ressourcentabelle *VFXRES.DBF*.

11.18. Benutzerspezifische Einstellungen

VFX erstellt für jedes Feld aus der Tabelle *VFXUSR.DBF* eine Public Variable mit dem Präfix *gu_* und erledigt vollautomatisch das Speichern und Lesen dieser Werte.

Nehmen wir an, dass Sie ein Feld mit dem Namen *TEST* in der Tabelle *VFXUSR* haben. Nach der Benutzeranmeldung wird eine Public Variable *gu_test* den Wert aus dem Feld *Test* der *VFXUSER*-Tabelle beinhalten. Falls diese Variable verändert wird, wird beim Verlassen der Anwendung dieser Wert wieder zurück in das Feld *Test* der Tabelle *VFXUSR* geschrieben.

Auf diese Weise ist es sehr einfach, benutzerspezifische Einstellungen zu speichern. Es reicht aus, in der Tabelle *VFXUSR* ein entsprechendes Feld anzulegen.

11.19. Include-Dateien

Die Include-Dateien spielen bei VFX eine wichtige Rolle. Es lohnt sich deshalb, die vorhandenen Include-Dateien etwas näher anzusehen:

Include-Datei	Verwendung	Ordner	Sprachabhängig?	Inhalt/Beschreibung
VFX.H	VFXMAIN.PRG	\VFX\INCLUDE\	Nein	Definiert die Konstanten _DEBUG_MODE, *LANG*SETUP, _DBCX und schließt andere Include-Dateien ein.
FOXPRO.H	VFX.H	Visual FoxPro Ordner	Nein	Standard-FoxPro-Definitionen.
VFXDEF.H	VFX.H	\VFX\INCLUDE\	Ja	Definiert die ID_LANGUAGE-Konstante und andere Konstanten.
VFXTXT.H	VFX.H	\VFX\INCLUDE\	Ja	Sprachabhängige Texte und Tooltip-Texte, die in der VFX-Entwicklungsumgebung verwendet werden.
VFXMSG.H	VFX.H	\VFX\INCLUDE\	Ja	Sprachabhängige Meldungstexte, die in der VFX-Entwicklungsumgebung verwendet werden.
VFXOFFCE.H	VFX.H	\VFX\INCLUDE\	Nein	In den Office-Klassen Word, Excel und Outlook verwendet.
USERTXT.H	VFX.H	\VFX\INCLUDE\	Ja	Sprachabhängige Texte und Tooltip-Texte, die Sie in Ihrer eigenen Anwendung verwenden. Die Datei wird von dem VFX – Message Editor erzeugt, wenn Sie die Option OTHER wählen.
USERMSG.H	VFX.H	\VFX\INCLUDE\	Ja	Sprachabhängige Meldungstexte, die Sie in Ihrer eigenen Anwendung verwenden. Die Datei wird von dem VFX - Message Editor erzeugt, wenn Sie die Option MESSAGE wählen.
USERDEF.H	VFX.H	\VFX\INCLUDE\	Nein	Sprachunabhängige Konstanten, die in Ihrer Anwendung verwendet werden.

Der VFX Anwendungs-Assistent generiert die meisten Konstanten automatisch, wenn Sie ein neues Projekt generieren. Wenn Sie den Debug-Modus oder die aktuelle Sprache wechseln wollen, müssen Sie Änderungen in einigen der Include-Dateien machen.

11.19.1. Define _Debug_Mode

VFX benutzt eine Konstante in der Include-Datei *VFX.H*, die angibt ob die Anwendung im Debug-Modus ablaufen soll oder nicht. Standardmäßig sind die folgenden Codezeilen in der Datei VFXMAIN.PRG, um den Debug-Modus in Abhängigkeit von der Konstanten *_DEBUG_MODE* einzustellen:

```
#ifdef _DEBUG_MODE
  goProgram.DebugMode(.t.)
#endif
```

Wenn Sie nicht wollen, dass Ihre Anwendung im Debug-Modus ausgeführt wird, kommentieren Sie Zeile mit der *_DEBUG_MODE*-Konstanten aus. Die Konstante befindet sich in der Include-Datei **VFX.H:**

```
…
* #DEFINE _DEBUG_MODE      .T.
…
```

11.19.2. Define ID_Language

In der Include-Datei *VFXDEF.H* ist die *ID_Language*-Konstante definiert, die die aktuelle Sprache Ihrer Anwendung angibt.

```
…
#define ID_LANGUAGE "ENG"
…
```

Wenn Sie Ihre Anwendung mit dem VFX-Anwendungs-Assistenten anlegen, wird die Anwendung in der Sprache angelegt, die im VFX-Anwendungs-Assistenten angegeben ist. Wenn Ihre Anwendung in eine andere Sprache übersetzt werden soll, ändern Sie die Konstante *ID_Language*. Lesen Sie im Kapitel *Mehrsprachige Anwendungen mit VFX* nach, um nähere Informationen zu erhalten.

11.19.3. Define _Lang_Setup

In der Include-Datei *VFX.H* gibt die Konstante *_LANG_SETUP* an, ob die *LangSetup()*-Methode ausgeführt wird oder nicht. Innerhalb der *LangSetup()*-Methode wird überprüft, ob die Konstante existiert. Nur wenn die Konstante existiert, wird der Code der *LangSetup()*-Methode ausgeführt. Dies dient der Geschwindigkeitsoptimierung in allen Formularen.

```
…
#DEFINE _LANG_SETUP        .T.
…
```

11.19.4. Kompilieren Ihrer Anwendung nach Änderungen in Include-Dateien

Um Visual FoxPro zu einem Neukompilieren zu veranlassen, müssen Sie eine Änderung in der oder den Datei(en) vornehmen, die die Include-Dateien einschließen. Der Befehl *clear program* im Befehlsfenster löscht alle kompilierten Programme im Hauptspeicher. Sie sollten die Datei *VFX.H* in Ihre Formulare einschließen, wenn Sie Konstanten in Ihren Formularen verwenden.

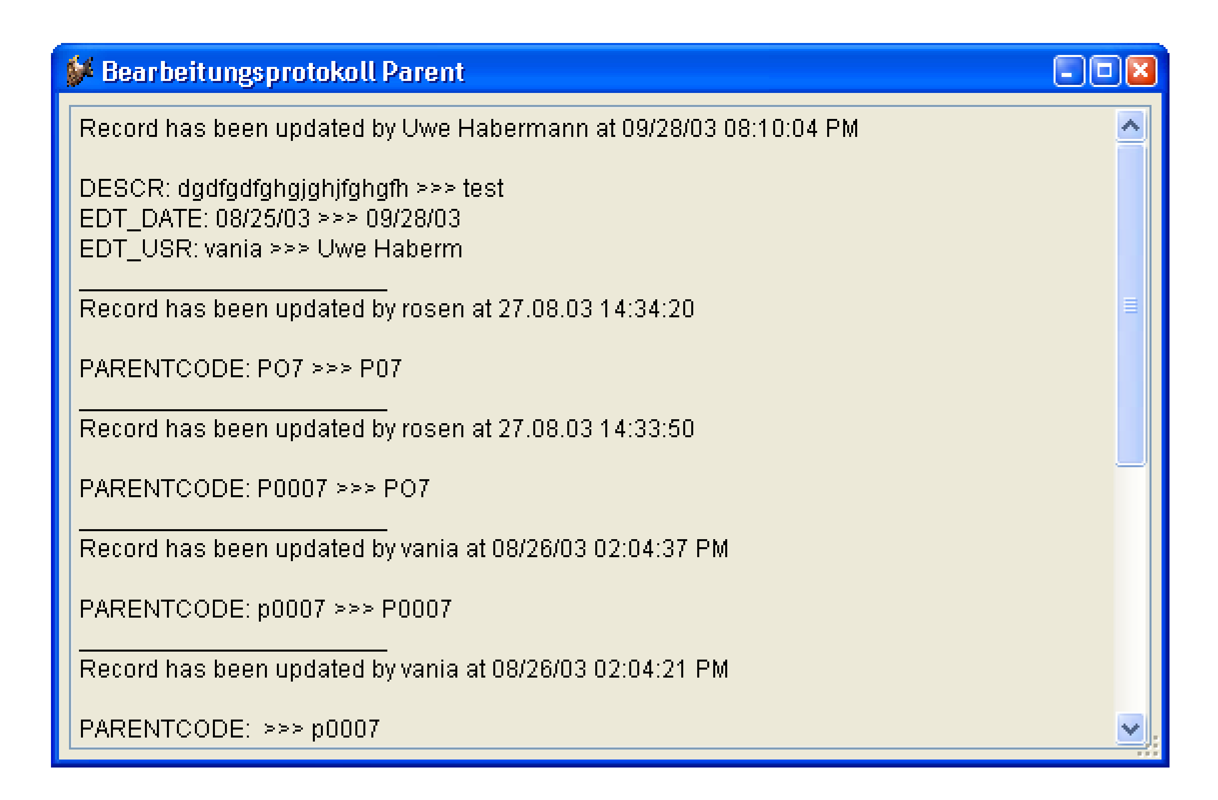
11.20. Bearbeitungsprotokoll

Das Bearbeitungsprotokoll (Audit-Trail) protokolliert Änderungen von Daten. VFX verwendet Trigger um die Änderung von Daten zu ermitteln. Die Trigger-Funktionen werden bei allen zu überwachenden Tabellen eingetragen.

- _audit_insert() protokolliert die Erfassung neuer Datensätze
- _audit_update() protokolliert alle Änderungen
- _audit_delete() protokolliert das Löschen von Datensätzen

Ein Audit-Trigger kann mit einem RI-Trigger mit einem logischen „und" verknüpft werden:

```
__ri_delete_parent() AND _audit_delete()
```



Über eine Schaltfläche in der Standard-Symbolleiste kann zum aktuell angezeigten Datensatz das Änderungsprotokoll angesehen werden.

11.21. OLE drag & drop

In VFX-Anwendungen steht OLE drag & drop auf drei verschiedene Arten zur Verfügung. Standardmäßig ist OLE drag & drop in Datenrastern eingeschaltet. Der gesamte Inhalt eines Datenrasters kann mit einem Mausklick zum Beispiel nach Excel kopiert werden.

Auf Wunsch können auch die Inhalte einzelner Steuerelemente per OLE drag & drop verschoben werden. Diese Eigenschaft ist standardmäßig ausgeschaltet und kann durch die Zeile:

```
nOLEenableDrag=1 && 0 use form setting (default), 1 enable, 2 disable
```

in Vfxmain.prg eingeschaltet werden.

Weiterhin ist es möglich die Daten aller Steuerelemente einer Seite eines Seitenrahmens in eine andere OLE drag & drop-fähige Anwendung zu kopieren. Auch diese Eigenschaft ist standardmäßig ausgeschaltet und kann bei Bedarf durch die Zeile:

```
nPageOLEdragdrop=1 && 0 use form setting (default), 1 enable, 2 disable
```

in Vfxmain.prg eingeschaltet werden.

11.22. Multi-Client-Support

Standardmäßig arbeitet eine VFX-Anwendung mit genau einer Datenbank, so wie es im VFX – Application Wizard eingetragen wurde. Auf Wunsch kann eine Mandantenfähigkeit eingebaut werden. Dazu ist in Vfxmain.prg der Datenpfad auf einen Leerstring zu setzen.

```
cdatadir = ''
```

Wenn der Datenpfad leer ist, sucht die VFX-Anwendung zur Laufzeit nach der Tabelle Vfxpath.dbf. Diese Tabelle muss sich im gleichen Ordner wie die ausführbare Programmdatei befinden. Wenn in dieser Tabelle genau ein Datensatz enthalten ist, wird der dort eingetragene Datenpfad verwendet. Enthält die Tabelle mehr als einen Datensatz erscheint beim Start der Anwendung ein Dialog zur Auswahl der gewünschten Datenbank.



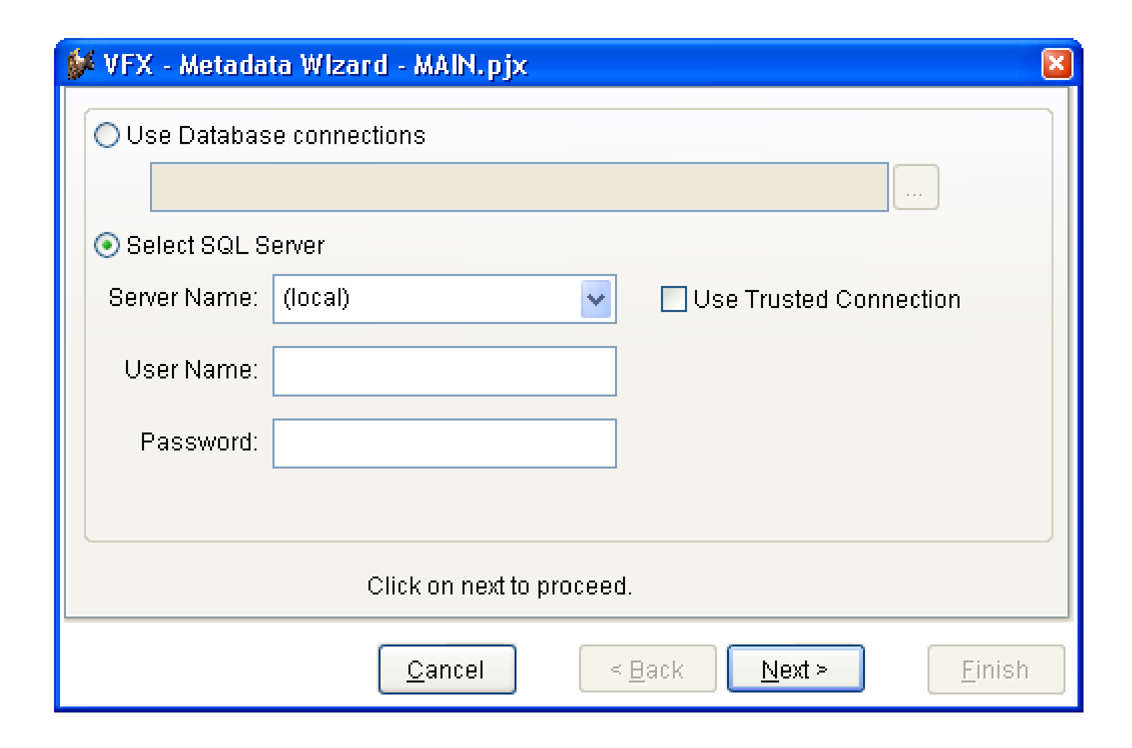
11.23. Aktualisierung der Kundendatenbank

11.23.1. Verwendung von VFP-Datenbanken

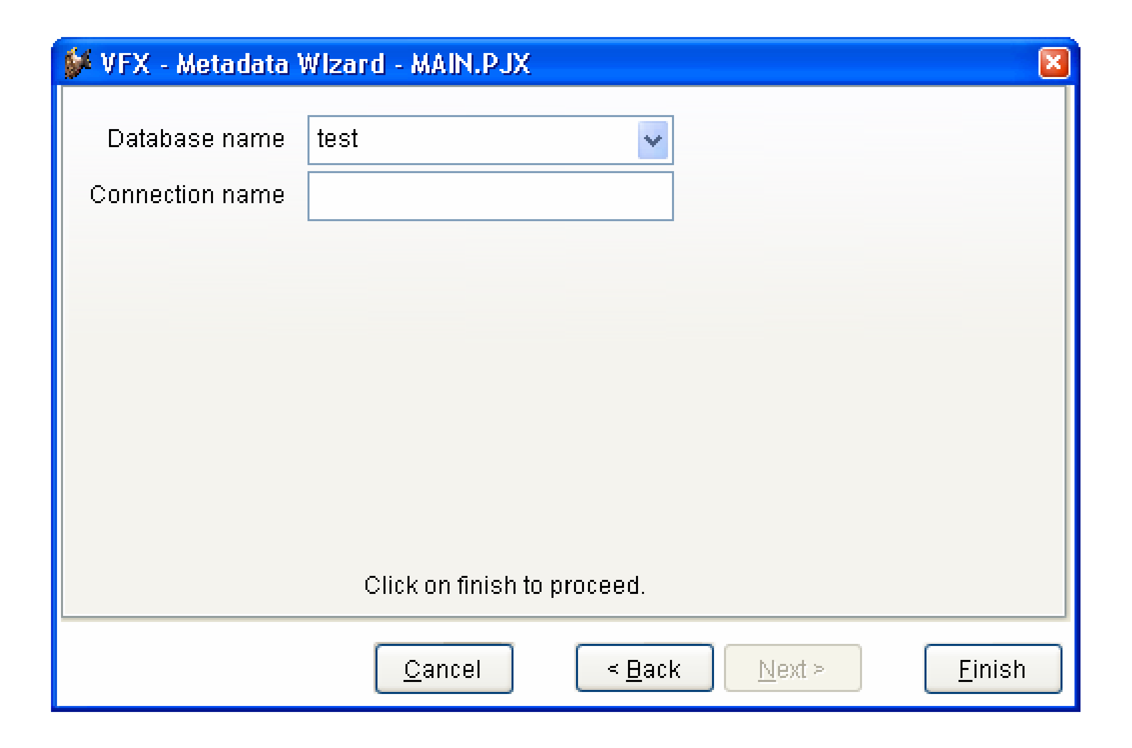
VFX bietet die Möglichkeit, die Datenbank beim Kunden automatisch zu aktualisieren. Die zu aktualisierenden Tabellen werden hierbei ohne Daten in den Update-Ordner unterhalb des Datenordners kopiert. Beim ersten Programmstart erkennt die VFX-Anwendung das Vorhandensein der Tabellen im Update-Ordner und aktualisiert die Datenbank. Es können auch freie Tabellen aktualisiert werden.

11.23.2. Verwendung von SQL Server-Datenbanken

Der Metadata Wizard hilft Ihnen Metadaten aus Ihrer aktuell benutzten SQL Server-Datenbank zu erstellen. Die Metadaten können zur Aktualisierung der Datenbank beim Kunden verwendet werden.



Wahlweise kann die Verbindung aus einer VFP-Datenbank ausgelesen werden um die Verbindung zu einem SQL Server herzustellen oder der SQL Server kann manuell ausgewählt werden.



Der Metadata Wizard erstellt die Tabelle „Datadict.dbf". Dies ist eine freie Tabelle, in der die Struktur der SQL Server Datenbank inklusiv Constraints, benutzerdefinierten Datentypen, Regeln, Ansichten und gespeicherten Prozeduren gespeichert wird. Der Metadata Wizard durchsucht das aktive Projekt nach Verbindungen und analysiert die Struktur der Datenbank. Wenn die Tabelle „Datadict.dbf" an die Kunden weitergegeben wird, wird die Struktur der dortigen Datenbank aktualisiert. Dabei wird wieder die bestehende Verbindung zum Zugriff auf die Datenbank verwendet.

11.24. Hooks

Eine elegante Möglichkeit in den Funktionsablauf von VFX-Methoden einzugreifen, ohne die Klassen verändern zu müssen, ist der Einsatz von Hooks.

Das Konzept der Hooks wurde in VFX 8.0 erweitert. Bisher war es möglich durch einen Hook innerhalb einer VFX-Methode einen eigenen Codeblock auszuführen. Über den Rückgabewert des Hooks konnte man steuern, ob der noch folgende VFX-Code in der Methode weiter ausgeführt werden sollte oder nicht. Der Rückgabewert, den die VFX-Methode dabei lieferte, konnte nicht beeinflusst werden und war in VFX fest vorgegeben.

Mit den erweiterten Hooks in VFX 8.0 kann nun zusätzlich der Rückgabewert der Methode vom Hook gesteuert werden.

Hooks sind in der Datei Vfxhook.prg gespeichert. Die Verwendung von Hooks kann im VFX – Application Manager oder in Vfxmain.prg mit der Zeile

```
nenablehook = 1
```

eingeschaltet werden. Nenablehook ist eine Eigenschaft des Applikationsobjekts.

Im folgenden Beispiel wird bei allen Steuerelementen, die disabled sind, die Schriftfarbe schwarz eingestellt.

```
function EventHookHandler(tcEvent, toObject, toForm)
   local lContinue
   lContinue = .T.
   DO CASE
   CASE UPPER(tcEvent)=="INIT"
      IF PEMSTATUS(toObject,"disabledforecolor",5)
         toObject.disabledforecolor=;
            eval(left(rgbscheme(1,2), ;
            at(",",rgbscheme(1,2),3)-1)+")")
         IF  PEMSTATUS(toObject,"disabledbackcolor",5)
            toObject.disabledbackcolor=;
               eval("rgb("+substr(rgbscheme(1,2),;
               at(",",rgbscheme(1,2),3)+1))
         ENDIF
      ENDIF
   ENDCASE
   return lContinue
endfunc
```

11.25. Hilfe bei der Fehlersuche

Fehler „cap_application_title not found“: Eine Include-Datei wurde nicht gefunden. Stellen Sie sicher, dass der aktuelle Ordner der Ordner Ihres Projektes ist! Tipp: Geben Sie folgenden Befehl im Befehlsfenster ein: *CD ?*. Beenden Sie VFP, starten Sie VFP erneut, setzen Sie den aktuellen Pfad auf Ihren Projektordner, öffnen Sie Ihr Projekt, wählen Sie “Alle Dateien nochmals kompilieren” und starten Sie anschließend Ihr Projekt. Hinweis: Wählen Sie die Option „Eigenschaften“ (letzte Option im Kontextmenü bei der Bearbeitung einer PRG-Datei) und wählen Sie „Vor dem Speichern kompilieren“. Dadurch haben Sie immer kompilierte PRG-Dateien.

Änderungen in den Include-Dateien werden nicht übernommen: Machen Sie eine Änderung in der Datei, die die Include-Datei einschließt, beenden Sie Visual FoxPro, löschen Sie alle kompilierten *FXP*-Dateien, starten Sie VFP erneut, wechseln Sie in den Projektordner und erstellen Sie das Projekt erneut. Tipp: Versuchen Sie auch den CLEAR PROGRAM-Befehl einzusetzen, der alle kompilierten Programme aus dem Speicher entfernt. Wenn Sie eine Änderung in einer Include-Datei machen, die von einem Formular eingeschlossen wird, öffnen Sie das Formular und speichern Sie es, sonst werden die Änderungen in der Include-Datei von dem Formular nicht übernommen. Wenn die Änderungen in Ihrer Include-Datei immer noch nicht wirksam werden, löschen Sie alle FXP-Dateien Ihres Projektes und wählen Sie „Alle Dateien neu kompilieren“.

Wichtig! Aktueller Ordner: Stellen Sie sicher, dass der aktuelle Ordner der Ordner mit dem Projekt ist, mit dem Sie arbeiten! Versuchen Sie: *CD ?*. **ANMERKUNG: Bevorzugen Sie die VFX Task Pane um Ihre Projekte zu öffnen.**

Erstellte Formulare basieren nicht auf Bibliotheken aus dem Ordner meiner Anwendung: Dies ist nur dann ein Problem, wenn Sie gleichzeitig an verschiedenen Projekten oder an verschiedenen Versionen eines Projektes arbeiten. Um fehlerhafte Verweise zu beseitigen, benennen Sie vorübergehend den Ordner Ihres Projektes um. Öffnen Sie alle Formulare und Klassen und wählen Sie, falls erforderlich, die richtige Klassenbibliothek für Ihre Anwendung und speichern Sie die Änderungen.

Inkrementelle Suche und andere VFX-Grid-Eigenschaften funktionieren nicht: Stellen Sie sicher, dass Sie den VFX - CGrid Builder, wie in diesem Handbuch beschrieben, verwenden.

Die Eigenschaft „inkrementelle Suche" steht nicht zur Verfügung: Sie müssen den Puffermodus auf 3 setzen, da sonst keine IDX-Dateien angelegt werden können. Möglicherweise steht der Puffermodus bei Ihnen auf 5!

1:n-Formular zeigt die Daten der Child-Tabelle nicht an, wenn ich den Datensatzzeiger der Haupttabelle bewege: Prüfen Sie, ob Sie die 1:n-Beziehung in der Datenumgebung des Formulars richtig eingestellt haben! Sie müssen nur per drag & drop eine Beziehung vom Primärschlüssel der Haupttabelle zum Fremdschlüssel der Child-Tabelle ziehen. Ändern Sie keine anderen Eigenschaften. **Tipp: Setzen Sie nicht die OneToMany-Eigenschaft** Ihrer 1:n-Beziehung in der Datenumgebung Ihres Formulars auf wahr. Das Setzen dieser Eigenschaft auf wahr entspricht der Ausführung des SET SKIP TO-Befehls. Dieses Verhalten ist an dieser Stelle NICHT erwünscht.

Die Auswahlliste funktioniert nicht mit numerischen Feldern: Setzen Sie die Eigenschaft *cReturnExpr* der *CPickField*-Klasse auf *TRANSFORM(Feld)* anstatt auf *Feld*. Alles weitere funktioniert genauso wie bei Zeichenfeldern.

Änderungen in *PRG*-Dateien wirken sich nicht aus: Führen Sie den Befehl CLEAR PROGRAM aus und versuchen Sie es erneut. Oder setzen Sie besser die Bearbeitungsoption auf „Vor dem Speichern kompilieren".

Fehler beim Neuerstellen eines Projektes: Wenn Sie Probleme beim Neuerstellen eines Projektes haben, wählen Sie die „Rebuild"-Option aus der VFX Task Pane wie oben beschrieben. **ANMERKUNG:** Die Include-Dateien und die Menüdateien sollten Sie von Hand überprüfen! Erwarten Sie nicht eine deutsche Anwendungsversion, wenn die Include-Dateien englisch sind.

11.26. Benutzen Sie die gewünschte Standard-Symbolleiste

Es ist vernünftig, für die Bedürfnisse Ihrer Anwendung (oder Ihrer Firma) eine eigene Klassenbibliothek anzulegen. Wir haben eine Klassenbibliothek mit dem Namen *APPL.VCX* für Sie vorbereitet. Um Ihnen die Arbeit so einfach wie möglich zu machen, haben wir in dieser Klassenbibliothek bereits zwei Klassen für Symbolleisten angelegt:

CAppBar und *CAppNavBar*.

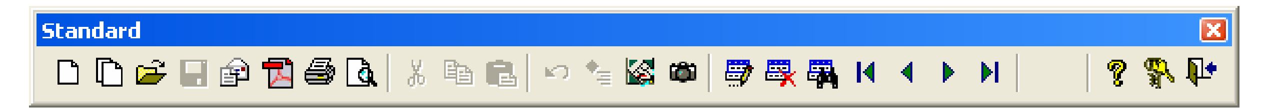
Die Erste ist die Standard-Symbolleiste und die Zweite ist eine Symbolleiste, die Sie verwenden können, wenn Sie Navigations- und andere Schaltflächen nicht auf Ihren Formularen haben wollen.

CAppBar:



CAppBar wird benutzt, wenn die Schaltflächen zur Navigation und zur Bearbeitung auf Ihren Formularen sind.

CAppNavBar:



CAppNavBar wird benutzt, wenn die Schaltflächen zur Navigation und zur Bearbeitung nicht auf Ihren Formularen sind.

Um zwischen diesen beiden Symbolleisten zu wechseln, brauchen Sie nur eine Eigenschaft der Anwendungsklasse in *VFXMAIN.PRG* zu ändern:

```
define class CApplication as CFoxApp

     …
     …
     ************************************************************
     ** CAppToolBar   - Tooolbar without Navigation Buttons
     ** CAppNavBar    - Tooolbar with    Navigation Buttons

     cMainToolBar = "CAppNavBar"
     …
     …
     …
```

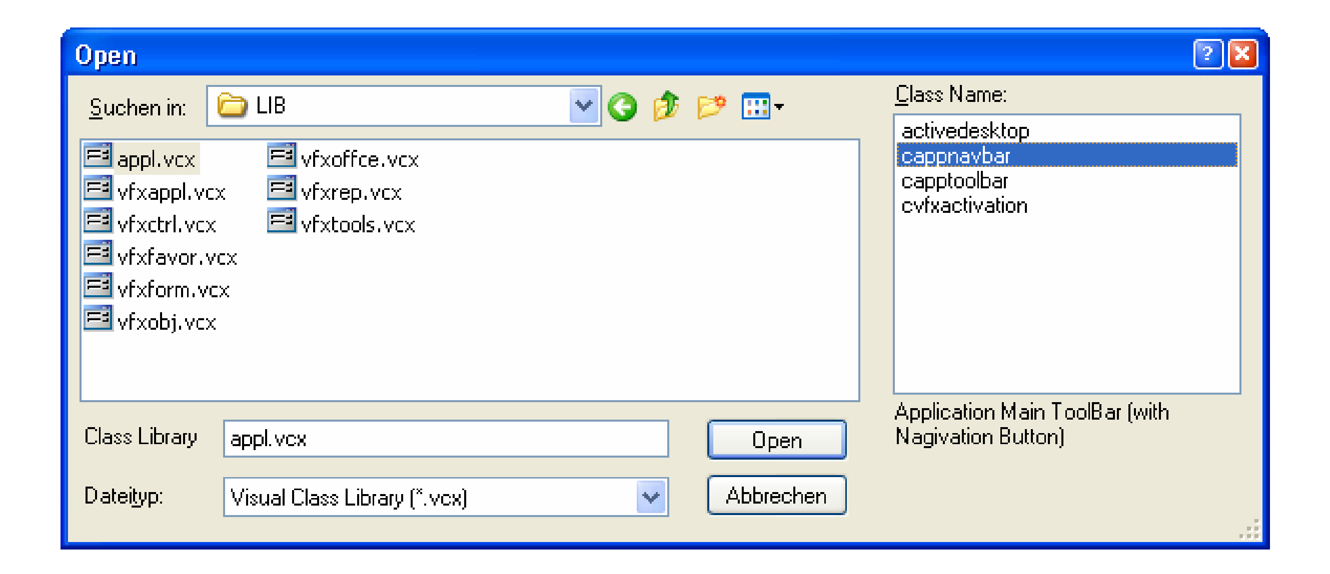
11.27. Erstellen Ihrer eigenen Symbolleistenklasse

Sie können die *CAppBar*- oder die *CAppNavBar*-Symbolleistenklassen für die meisten Office-kompatiblen Anwendungen benutzen. Aber selbstverständlich können Sie auch andere Symbolleisten verwenden. Sie müssen nur eine neue Klasse erstellen, die von der *CToolbar*-Klasse oder auch von der *CAppBar*- oder der *CAppNavBar*-Klasse vererbt wird. Hier wird gezeigt wie es geht:

Wählen Sie *Neu,* wenn Sie sich auf der Klassenseite des Projekt-Managers befinden. Es wird folgendes Dialogfenster angezeigt:

Class Name: Geben Sie den Namen der neuen Klasse ein. Wir nennen sie hier *CMyToolbar*.

Based On: Drücken Sie auf die Schaltfläche mit den drei Punkten und das folgende Dialogfenster wird geöffnet. Wählen Sie die Klasse *CAppBar* (oder *CAppNavBar*) aus der VFX-Klassenbibliothek *APPL.VCX*.



From: Die Referenz auf die VFX-Klassenbibliothek mit dem Namen *APPL.VCX* wird automatisch angezeigt.

Store In: Wenn Ihre anwendungsspezifische Klassenbibliothek noch nicht existiert, geben Sie den vollständigen Pfadnamen an. Andernfalls wählen Sie Ihre Klassenbibliothek mit der Schaltfläche mit den drei Punkten (Dialog zur Dateiauswahl).

11.28. Anpassen der Symbolleistenklasse

Jetzt müssen Sie Ihre Symbolleistenklasse anpassen. Sie machen dies mit dem Klassen-Designer.

11.28.1. Einen Zwischenraum einfügen

Fangen Sie mit einem Zwischenraum an, um die erste anwendungsspezifische Schaltfläche von der letzten Schaltfläche der Standard-Symbolleiste zu trennen.



Benutzen Sie dieses Symbol aus der Visual FoxPro Symbolleiste für Formular-Steuerelemente und ziehen Sie es auf Ihre Symbolleiste wo es benötigt wird.

11.28.2. Eine Schaltfläche einfügen

Visual Extend bietet vordefinierte Schaltflächen für die einfache Erstellung von Symbolleisten. Ziehen Sie die Klasse *cToolbarbutton* aus der VFX-Klassenbibliothek *VFXCTRL.VCX* auf Ihre Symbolleiste und passen Sie die folgenden Eigenschaften und Methoden an Ihre Bedürfnisse an:

Click Event: Tragen Sie die Befehle ein, die immer dann ausgeführt werden sollen, wenn der Benutzer auf diese Schaltfläche drückt. Wenn Sie beispielsweise das Formular *Customer* öffnen wollen, geben Sie folgenden Code:

```
goProgram.RunForm("CUSTOMER")
```

in die Click()-Methode ein.

Picture: Wählen Sie eine *BMP*- oder *ICO*-Datei aus, die als Beschriftung Ihrer Schaltfläche angezeigt wird.

ANMERKUNG: Fügen Sie den folgenden Code in das *Refresh()*-Ereignis jeder Schaltfläche oder Ihrer Symbolleiste ein. Sie stellen damit sicher, dass die Schaltflächen immer richtig angezeigt werden. Wenn Sie ein modales Formular öffnen, wird VFX die Schaltflächen in den Symbolleisten deaktivieren. Sie können mit folgendem Code sicherstellen, dass die Schaltflächen wieder richtig aktiviert werden:

```
this.enabled = this.parent.cmdopen.enabled
```

Mit diesem Code wird die Schaltfläche der Symbolleiste automatisch mit dem Anzeigeverhalten der Schaltfläche *Öffnen* synchronisiert.

11.28.3. Beispiel einer anwendungsspezifischen Symbolleiste



11.29. Symbolleisten zu Formularen

Es hat sich als sehr praktisch erwiesen Formularen eigene Symbolleisten zuordnen zu können. Die Symbolleisten sollten auf der Klasse *ctoolbar* basieren und in der Klassenbibliothek Appl.vcx gespeichert werden. Der Name der Symbolleiste wird dem Formular in der Eigenschaft *ctoolbarclass* bekannt gemacht.

VFX instanziiert die Symbolleiste zusammen mit dem Formular. Die Symbolleiste ist sichtbar, solange das Formular das aktive Formular ist.

Um zum Beispiel ein Child-Formular über eine Schaltfläche in einer Symbolleiste zu öffnen, fügen wir der Symbolleiste eine Schaltfläche basierend auf der Klasse *ctoolbarclass* hinzu. In das Click-Event der Schaltfläche schreiben wir:

```
_screen.activeform.onmore(1)
```

Das ist alles. Da VFX sicherstellt, dass die Symbolleiste nur dann sichtbar ist, wenn das dazugehörige Formular aktiv ist, können wir sicher sein, dass _screen.activeform existiert. Von diesem Formular wird die onmore Methode aufgerufen und bekommt als Parameter eine 1 übergeben. Damit wird das Formular aufgerufen, das im ersten Array-Element der onmore Methode angegeben ist, ohne dass der OnMore-Dialog angezeigt wird.

11.30. Eigenschaften der Klasse cApplication

Die Klasse cApplication ist die Klasse des Applikationsobjekts. Die Eigenschaften und Methoden des Applikationsobjekts stehen global in der gesamten Anwendung zur Verfügung.

Die Klasse cApplication wird in Vfxmain.prg programmatisch von der visuellen Klasse cFoxapp aus der Klassenbibliothek Vfxappl.vcx abgeleitet. Die Werte der Eigenschaften können in Vfxmain.prg unter DEFINE CLASS capplication AS cFoxapp eingestellt werden. Ebenso können hier Methoden der Klasse vererbt und überschrieben oder verändert werden. Die für die Steuerung der Anwendung wichtigen Eigenschaften des Applikationsobjekts sollen hier im Einzelnen erläutert werden.

cAscOrderRGB – RGB-Wert einer Farbe, die verwendet wird um eine aufsteigende Sortierung in einer Grid-Spalte in der Überschrift anzuzeigen. Der Standardwert ist *"RGB(255,255,0)"*.

cDataDir - Der Pfad in dem sich die Datenbank befindet. Standardmäßig wird dieser Pfad aus der Konstanten *datapath_loc* aus der Datei *Userdef.h* gelesen. Weisen Sie dieser Eigenschaft einen Leerstring zu, wenn Sie Multi-Client-Database-Eigenschaft von VFX nutzen möchten. In diesem Fall sind der Tabelle *Vfxpath.dbf* Datensätze hinzuzufügen, die auf die entsprechenden Datenpfade verweisen.

cDateFormat – Das Datumsformat, das standardmäßig in der Applikation verwendet wird. Der Wert dieser Eigenschaft wird als Parameter dem Befehl SET DATE übergeben. Der Wert dieser Eigenschaft wird normalerweise in der Methode *setlangid* des Applikationsobjekts entsprechend der eingestellten Sprache zugewiesen.

cDescOrderRGB – RGB-Wert einer Farbe, die verwendet wird um eine absteigende Sortierung in einer Grid-Spalte in der Überschrift anzuzeigen. Der Standardwert ist *"RGB(255,0,0)"*.

cEdt_Date – Der Name eines Feldes in einer beliebigen Tabelle. Wenn ein Datensatz gespeichert wird und in der betreffenden Tabelle ein Feld mit diesem Namen gefunden wird, werden hier das Datum und ggf. die Uhrzeit der Bearbeitung gespeichert. Der Typ des Feldes kann Date oder Datetime sein. Der Standardwert ist ein Feld mit dem Namen *edt_date*.

cEdt_Usr – Der Name eines Feldes in einer beliebigen Tabelle. Wenn ein Datensatz gespeichert wird und in der betreffenden Tabelle ein Feld mit diesem Namen gefunden wird, wird hier der Name des Benutzers gespeichert, der den Datensatz verändert hat. Das Feld muss vom Typ Zeichen sein. Der Standardwert ist ein Feld mit dem Namen *edt_usr*.

cExcludeFiles – Hier kann eine durch Komma separierte Liste von Dateinamen eingegeben werden. Die hier aufgeführten Dateien erscheinen nicht im Dialog Datenbankwartung und sind von der Datenbankwartung ausgeschlossen. Der Standardwert ist *"DBCXREG.DBF;CDBKMETA.DBF;SDTMETA.DBF;SDTUSER.DBF;COREMETA.DBF"*.

cHelpFile – Der Name der Hilfedatei, die beim drücken der Taste F1 geöffnet werden soll. Die Namenserweiterung (*chm* oder *hlp*) muss mit angegeben werden. Der Standardwert ist der Name des Projekts mit der Namenserweiterung *chm*.

cIns_Date – Der Name eines Feldes in einer beliebigen Tabelle. Wenn ein neuer Datensatz gespeichert wird und in der betreffenden Tabelle ein Feld mit diesem Namen gefunden wird, werden hier das Datum und ggf. die Uhrzeit der Neuanlage gespeichert. Der Typ des Feldes kann Date

Date oder Datetime sein. Der Standardwert ist ein Feld mit dem Namen *ins_date*.

cIns_Usr – Der Name eines Feldes in einer beliebigen Tabelle. Wenn ein neuer Datensatz gespeichert wird und in der betreffenden Tabelle ein Feld mit diesem Namen gefunden wird, wird hier der Name des Benutzers gespeichert, der den Datensatz neu angelegt hat. Das Feld muss vom Typ Zeichen sein. Der Standardwert ist ein Feld mit dem Namen *ins_usr*.

cIntroBitmap – Der Name einer Bilddatei, die als Splashscreen angezeigt werden soll. Es sind alle von VFP unterstützten Grafikformate zulässig, also zum Beispiel *bmp*, *jpg*, *gif* oder *png*. Der Standardwert ist *Bitmap\Intro.png* und wird aus der Include-Datei *Userdef.h* gelesen.

cIntroForm – Der Name einer Formularklasse, die den Splashscreen anzeigen soll. Eine Änderung dieses Wertes ist nur erforderlich, wenn ein Splashscreen mit besonderen Eigenschaften verwendet werden soll. Der Standardwert ist *cSplashDialog*.

cLoginForm – Der Name einer Formulardatei, die den Anmeldedialog enthält. Eine Änderung dieser Eigenschaft ist nur erforderlich, wenn die Benutzerverwaltung von VFX nicht verwendet soll und ein eigenes Verfahren zur Benutzerverwaltung zum Einsatz kommt. Der Standardwert ist *Vfxlogin.scx*.

cMainDatabase – Der Name der Datenbank. Der Wert wird aus der Konstanten *database_loc* aus der Datei *Userdef.h* gelesen. Der Standardwert wurde mit dem VFX – Application Wizard beim Erstellen des Projekts festgelegt.

cMainForm – Der Name eines Formulars, das beim Start der Anwendung nach der Benutzeranmeldung angezeigt werden soll. Der Standardwert ist eine leere Zeichenkette.

cMainIcon – Das Symbol der Anwendung. Standardmäßig wird dieses Symbol in allen Formularen verwendet. Der Standardwert ist *Bitmap\Main.ico* und wird aus der Konstanten *mainicon_loc* aus der Include-Datei *Userdef.h* gelesen

cMainTitle – Der Name der Anwendung. Dieser Name erscheint in der Titelzeile der Anwendung. Der Name der Anwendung kann auch beim Befehl *CREATEOBJECT(„capplication“,<Name der Anwendung>)* als zweiter Parameter angegeben werden. In diesem Fall wird der Wert dieser Eigenschaft überschrieben. Der Standardwert ist *Untitled*.

cMainToolbar – Der Name der Standard-Symbolleiste. Der Standardwert wurde mit dem VFX – Application Wizard beim Anlegen des Projekts

festgelegt. VFX stellt zwei Klassen mit Symbolleisten zur Verfügung. Die Klasse *CAppToolbar* enthält keine Schaltflächen zur Bewegung des Datensatzzeigers in Formularen. Die Klasse *CAppBavBar* enthält Schaltflächen zur Bewegung des Datensatzzeigers in Formularen.

cvfxpath – In dieser Eigenschaft kann der Name der Tabelle angegeben werden, die die Informationen zu den Pfaden der Datenbanken der Anwendung enthält. Der Standardwert ist *Vfxpath.dbf*.

FileMnuName – In dieser Eigenschaft wird der Name des Menü-Pads „Datei" eingegeben. Der Name muss nicht der angezeigten Überschrift entsprechen. In *Vfxmenu.vmx* wird der Name *File* verwendet. Der Name muss dem Applikationsobjekt bekannt sein, weil diesem Menü-Pad zur Laufzeit Einträge für die zuletzt verwendeten Formulare hinzugefügt werden.

FileMnuOffset – Dies ist die Nummer des Eintrags im Menü-Pad „*Datei*", das für den ersten Eintrag eines zuletzt verwendeten Formulars verwendet wird. Wenn Sie dem Menü „*Datei*" eigene Einträge hinzufügen wollen, muss dieser Wert entsprechend erhöht werden.

lAllowDeleteChildData - Wenn der Wert dieser Eigenschaft auf *wahr* gesetzt wird, dürfen Benutzer, die in OneToMany-Formulare keine Datensätze löschen dürfen, trotzdem Child-Datensätze löschen. Wenn dieser Wert auf *falsch* gesetzt wird, dürfen auch keine Child-Datensätze gelöscht werden.

lAutoLogin – Wenn der Wert dieser Eigenschaft auf *wahr* gesetzt wird, werden Benutzer, die in der Benutzerverwaltung registriert sind, beim Anwendungsstart ohne Aufforderung zur Eingabe eines Kennworts automatisch angemeldet. Der Standardwert dieser Eigenschaft ist *falsch*.

lCentury – Wenn der Wert dieser Eigenschaft auf wahr gesetzt ist, wird in allen Datumsfeldern der Anwendung die Jahreszahl 4stellig angezeigt. Der Standardwert ist *falsch*, Jahreszahlen werden 2stellig angezeigt.

lDisableFormResize – Wenn der Wert dieser Eigenschaft auf *wahr* gesetzt wird, ist das Ändern der Größe aller Formulare der Anwendung nicht möglich. Der Standardwert ist *falsch*, die Größe von Formularen kann vom Benutzer verändert werden.

lNoClearIdxOnDelete – Standardmäßig löscht VFX temporäre Indexdateien, wenn ein Datensatz gelöscht werden soll. Setzen Sie den Wert dieser Eigenschaft auf *wahr*, wenn temporäre Indexdateien in dieser Situation nicht gelöscht werden sollen. Beachten Sie, dass temporäre Indexdateien nicht geöffnet sein dürfen, wenn Transaktionen ausgeführt werden. Der Standardwert ist *falsch*.

lNoClearIdxOnEdit – Standardmäßig löscht VFX temporäre Indexdateien, wenn ein Datensatz bearbeitet werden soll. Setzen Sie den Wert dieser Eigenschaft auf *wahr*, wenn temporäre Indexdateien in dieser Situation nicht gelöscht werden sollen. Beachten Sie, dass temporäre Indexdateien nicht geöffnet sein dürfen, wenn Transaktionen ausgeführt werden. Der Standardwert ist *falsch*.

lNoClearIdxOnInsert – Standardmäßig löscht VFX temporäre Indexdateien, wenn ein Datensatz neu angelegt werden soll. Setzen Sie den Wert dieser Eigenschaft auf *wahr*, wenn temporäre Indexdateien in dieser Situation nicht gelöscht werden sollen. Beachten Sie, dass temporäre Indexdateien nicht geöffnet sein dürfen, wenn Transaktionen ausgeführt werden. Der Standardwert ist *falsch*.

lRelogonQuit – Steuert das Verhalten der Anwendung, wenn ein Benutzer versucht sich während die Anwendung läuft erneut anzumelden und den Vorgang abbricht. Wenn der Wert dieser Eigenschaft auf *wahr* gesetzt wird, wird die Anwendung beendet. Wenn der Wert dieser Eigenschaft auf *falsch* gesetzt wird, bleibt der zuletzt angemeldete Benutzer angemeldet.

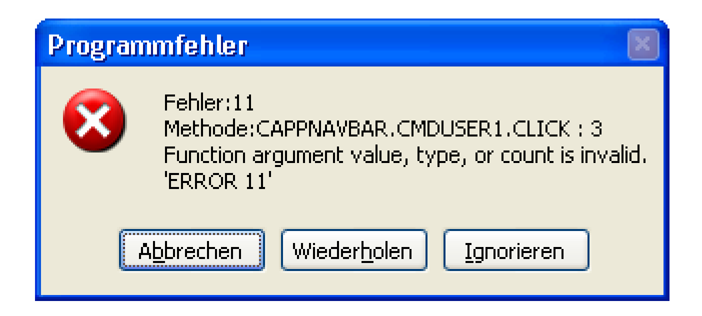
lRemakeIdxAfterClear – Wenn der Wert dieser Eigenschaft auf *wahr* gesetzt wird, werden temporäre Indexdateien nach dem Abschluss eines Speichervorgangs automatisch wieder angelegt. Vergleichen Sie auch mit den Eigenschaften *lNoClearIdxOnDelete*, *lNoClearIdxOnEdit*, *lNoClearIdxOnInsert*. Der Standardwert dieser Eigenschaft ist *falsch*.

In VFX 8.0 wurde die Klasse des Applikationsobjekts um eine Reihe neuer Eigenschaften zur Steuerung des Verhaltens der Applikation im Fehlerfall, zur Verwendung der Produktaktivierung und zur Installation eines Postscript-Druckertreibers, der zur Erstellung von PDF-Dateien benötigt wird, erweitert.

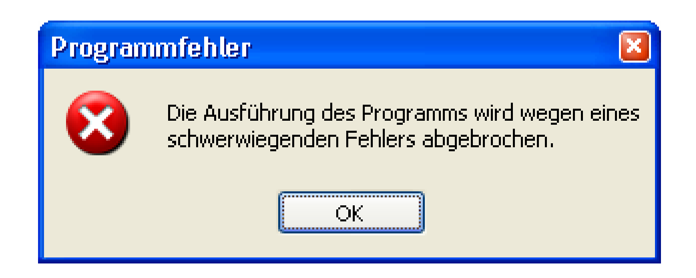
nAppOnErrorBehavior – Diese Eigenschaft steuert das Verhalten der Applikation im Fehlerfall.

0 – Laufzeitfehler werden ignoriert.

1 – Es wird eine Fehlermeldung angezeigt (Standardwert).



2 – Die Ausführung der Applikation wird nach Anzeige eines Hinweises beendet.



ErrorDetailLevel – Diese Eigenschaft steuert welche Informationen im Fehlerfall in der Tabelle *Vfxlog.dbf* protokolliert werden.

0 – Nur die Fehlermeldung aber keine Information über den Aufrufstapel.

1 – Die Fehlermeldung und Informationen über den Aufrufstapel (Standardwert).

2 – Vollständige, detaillierte Fehlerinformationen.

PSPrinterToInstall – Diese Eigenschaft enthält den Namen des Standard-Postscript-Druckertreibers. Dieser Druckertreiber wird automatisch installiert, wenn noch kein Postscript-Druckertreiber installiert ist und die Applikation einen Postscript-Druckertreiber braucht um eine PDF-Datei zu erstellen. Der Standardwert ist "HP DeskJet 1200C/PS".

cConnectionCheckURL – Diese Eigenschaft enthält die Adresse einer Internetseite, die verwendet wird um zu testen, ob eine Internet-Verbindung besteht. Diese Eigenschaft wird benötigt wenn Ghostscript nicht installiert ist. Ghostscript wird bei Bedarf automatisch aus dem Internet heruntergeladen und installiert. Ghostscript wird verwendet um Postscript-Dateien in PDF-Dateien umzuwandeln. Wenn keine Verbindung mit dem Internet besteht und auch keine DFÜ-Netzwerkverbindung eingerichtet ist, wird von VFX ein Eintrag im DFÜ-Netzwerk angelegt. Alle Eigenschaften der DFÜ-Verbindung

können vom Entwickler vorgegeben werden. Der Anwender kann bei Bedarf in einem Dialog die Telefonnummer, den Benutzernamen und das Kennwort ändern.

lUseActivation – Über diese Eigenschaft wird die Produktaktivierung ein- bzw. ausgeschaltet. Diese Eigenschaft kann im VFX - Application Wizard eingestellt werden, wenn ein neues Projekt erstellt wird. Später kann der Eintrag in Vfxmain.prg geändert werden. Der Standardwert ist .F., die Produktaktivierung wird nicht verwendet.

lActivationType – Wenn diese Eigenschaft auf .T. gesetzt wird, überprüft die Klasse cVFXActivate ob die Datei „FirstInstall.txt" existiert, wenn die Applikation gestartet wird. Diese Eigenschaft kann im VFX Application Wizard eingestellt werden, wenn ein neues Projekt erstellt wird. Später kann der Eintrag in Vfxmain.prg geändert werden. Der Standardwert ist .F., es wird nicht auf das Vorhandensein der Datei „FirstInstall.txt" geprüft.

11.31. Die Klasse cDownload

Diese Klasse ermöglicht das Herunterladen von Dateien aus dem Internet. Bei Bedarf können die heruntergeladenen Dateien ausgeführt werden und es können weitere Aktionen ausgeführt werden. Insbesondere ist hierdurch die Installation von Programmen aus dem Internet möglich.

Durch die Verwendung von Makros und die Execmacro-Funktion von VFP kann diese Klasse sehr vielseitig eingesetzt werden.

Makros sind Zeichenketten, die eine Folge von Befehlen aus der Makrosprache enthalten. Eigene Makros können erstellt werden. Ein Beispiel ist in der Tabelle Vfxsys.dbf im Feld *Install_GS* zu finden. Mit diesem Makro wird das Programm Ghostscript aus dem Internet heruntergeladen und installiert.

Diese Klasse verwendet die in der Eigenschaft *goProgram.cConnectionCheckURL* gespeicherte Internetseite um zu überprüfen, ob eine Internetverbindung besteht. Bei Bedarf wird eine Verbindung automatisch hergestellt. Wenn im DFÜ-Netzwerk keine Verbindung eingetragen ist, wird ein neuer Eintrag hinzugefügt. Die Verbindungsinformationen kann der Entwickler in den Eigenschaften vorgeben. Der Anwender kann die Telefonnummer, den Benutzernamen und das Kennwort für die neue Verbindung bei Bedarf in einem Dialog ändern.

11.31.1. Eigenschaften

LastErrorNo – Diese Eigenschaft enthält die Nummer des letzten Fehlers (falls ein Fehler aufgetreten ist). Damit kann die Ursache des letzten Fehlers ermittelt werden.

LastErrorTest – Wenn ein Fehler aufgetreten ist, ist in dieser Eigenschaft der Text der Fehlermeldung zu finden.

11.31.2. Methoden

ExecMacro (vcMacro, lnNoRun)

vcMacro – Skript der Makrosprache, das ausgeführt werden soll.

lnNoRun – Wenn diese Eigenschaft auf .T. gesetzt wird, wird die heruntergeladene Datei nicht ausgeführt.

11.31.3. Befehle der Makrosprache

„D:“ URL

Unter dieser Internetadresse ist die herunterzuladende Datei zu finden. Dieser Befehl führt die Datei nach dem erfolgreichen Herunterladen aus, wenn die Eigenschaft *lnNoRun* auf .F. gesetzt ist.

„C:“ nTimeOut; lPartial; lTopLevelForm; lResultOnError; SearchedString

Wartet bis das Fenster mit dem Titel *SearchedString* erscheint.

nTimeOut – Timeout in Sekunden. Wenn das erwartete Formular nicht innerhalb dieser Zeitspanne erscheint, wird ein Timeout-Fehler erzeugt.

lPartial – Wenn der Wert dieser Eigenschaft auf .T. gesetzt ist, reicht es, wenn der übergebene Titel einem Teil des Fensternamens entspricht. Wenn diese Eigenschaft auf .F. gesetzt ist, muss der übergebene Titel exakt dem Namen des Fensters entsprechen.

lTopLevelForm – Wenn der Wert dieser Eigenschaft auf .T. gesetzt ist, wird der Fenstername nur in Top-Level-Fenstern gesucht.

lResultOnError – Mit dieser Eigenschaft wird das Verhalten des Skripts gesteuert, falls das Fenster nicht innerhalb der vorgegebenen Zeitspanne gefunden wurde. Wenn das Fenster für die weitere Ausführung des Skripts zwingend erforderlich ist, muss nach Ablauf der vorgegebenen Zeitspanne die Ausführung des Skripts abgebrochen werden. In diesem Fall muss der Wert von *lResultOnError* auf .F. gesetzt werden. Wenn die Ausführung des Skripts unabhängig vom Vorhandensein

des Fensters nach der vorgegebenen Zeitspanne fortgesetzt werden soll, muss *lResultOnError* auf .T. gesetzt werden.

SearchedString

Bezeichnung, die in einem Fensternamen gesucht wird.

„W:" nTimeOut; lPartial; lTopLevelForm; lRezultByError; SearchedString

Es wird gewartet bis das Fenster, das die angegebene Zeichenkette im Titel enthält, geschlossen ist.

nTimeOut – Timeout in Sekunden. Wenn das erwartete Fenster innerhalb dieser Zeitspanne nicht geschlossen ist, wird ein Timeout-Fehler ausgelöst.

lPartial – Wenn der Wert dieser Eigenschaft auf .T. gesetzt ist, reicht es, wenn der übergebene Titel einem Teil des Fensternamens entspricht. Wenn diese Eigenschaft auf .F. gesetzt ist, muss der übergebene Titel exakt dem Namen des Fensters entsprechen.

lTopLevelForm – Wenn der Wert dieser Eigenschaft auf .T. gesetzt ist, wird der Fenstername nur in Top-Level-Fenstern gesucht.

lResultOnError – Mit dieser Eigenschaft wird das Verhalten des Skripts gesteuert, falls das Fenster nicht innerhalb der vorgegebenen Zeitspanne gefunden wurde. Wenn das Fenster für die weitere Ausführung des Skripts zwingend erforderlich ist, muss nach Ablauf der vorgegebenen Zeitspanne die Ausführung des Skripts abgebrochen werden. In diesem Fall muss der Wert von *lResultOnError* auf .F. gesetzt werden. Wenn die Ausführung des Skripts unabhängig vom Vorhandensein des Fensters nach der vorgegebenen Zeitspanne fortgesetzt werden soll, muss *lResultOnError* auf .T. gesetzt werden.

SearchedString

Eine Zeichenkette nach der im Titel eines Fensters gesucht wird.

„X:"

Schließt das Top-Level-Fenster. Mit dem „*C:*"-Befehl muss zuvor sichergestellt werden, dass das gewünschte Fenster sichtbar ist.

„K:" nKeyCode1; nKeyCode2; …

Die aufgeführten Tastenschlüssel werden in den Windows-Tastaturpuffer übertragen.

„U:" URL

Von dieser Internetadresse wird das Herunterladen ausgeführt. Die heruntergeladene Datei wird unabhängig vom Wert der Eigenschaft *lnNoRun* nicht ausgeführt.

11.31.4. Beispiel

Beschreibung der Installation von Ghostscript:

D: ftp://mirror.cs.wisc.edu/pub/mirrors/ghost/AFPL/gs811/gs811w32.exe

Lädt die Datei gs811w32.exe aus dem Internet herunter und führt sie anschließend aus.

C: 30; .F.; .F.; .F.; WinZip Self-Extractor - gs811w32.exe

Wartet bis das Fenster mit dem Titel „WinZip Self-Extractor - gs811w32.exe" erscheint.

K: 43

Sendet den Tastenschlüssel „Eingabetaste" an das aktive Fenster. Dadurch wird das Entpacken der Dateien ausgelöst.

C: 60; .F.; .F.; .F.; AFPL Ghostscript Setup

Wartet bis das Fenster mit dem Titel „AFPL Ghostscript Setup" erscheint.

K: 43

Sendet den Tastenschlüssel „Eingabetaste" an das aktive Fenster. Dadurch wird die Installation von Ghostscript gestartet.

W: 240; .F.; .F.; .F.; AFPL Ghostscript Setup Log

Wartet solange das Fenster „AFPL Ghostscript Setup Log" geöffnet ist. Dieses Fenster zeigt den Fortschritt der Installation an und die Skriptausführung muss warten, bis dieser Vorgang beendet ist.

C: 30; .T.; .T.; .T.; Ghostscript

Wartet bis das Fenster mit dem Titel „Ghostscript" erscheint. Dieses Fenster zeigt die Nachricht an, dass die Installation erfolgreich war.

X:

Schließt das letzte Fenster.

Hiermit ist die Installation von Ghostscript beendet.

11.32. Die Klasse cCreatePDF

Diese Klasse erstellt Berichtsausgaben im PDF-Format. Als Parameter werden der Aliasname des zu verwendenden Cursors, der Name der zu erstellenden PDF-Datei, der Name der Berichtsdatei sowie eine optionale For-Klausel übergeben.

Um eine PDF-Datei erstellen zu können, müssen Ghostscript und ein Postscript-Druckertreiber auf dem jeweiligen Computer installiert sein. Diese Klasse prüft, ob Ghostscript bereits installiert ist. Sollte dies nicht der Fall sein, wird Ghostscript automatisch aus dem Internet heruntergeladen und installiert. Für das Herunterladen aus dem Internet wird die Klasse cDownload verwendet. In dem Memofeld *Install_gs* aus der Tabelle Vfxsys.dbf befindet sich das Skript, das zum Herunterladen und zur Installation von Ghostscript verwendet wird. In der Beschreibung der Klasse cDownload befinden sich weitere Hinweise.

Wenn kein Postscript-Druckertreiber installiert ist, installiert diese Klasse automatisch den Druckertreiber, dessen Name in der Eigenschaft *goProgram.PSPrinterToInstall* hinterlegt ist. In der Regel sind hierfür keine Benutzereingaben erforderlich.

Der Bericht wird über den Postscript-Druckertreiber ausgegeben und in einer Datei gespeichert. Das Programm Ghostscript wandelt diese Postscript-Datei in eine PDF-Datei um.

11.32.1. Eigenschaften

LastErrorNo – Diese Eigenschaft enthält die Nummer des letzten Fehlers (falls ein Fehler aufgetreten ist). Damit kann die Ursache des letzten Fehlers ermittelt werden.

LastErrorTest – Wenn ein Fehler aufgetreten ist, ist in dieser Eigenschaft der Text der Fehlermeldung zu finden.

11.32.2. Methoden

AddAttachment (tsAlias, tcFileName, tcReport, tcFor)

Fügt dem CreatePDF-Objekt Informationen über eine Datei hinzu. Es wird automatisch eine PDF-Datei zum dem als Parameter übergebenen Bericht erstellt. Ein weiterer Ausdruck kann als Parameter angeben werden. Dieser Ausdruck dient dazu die Daten des Berichts zu filtern.

Create_PDF(tcAlias, tcRezFile, tcFRXName, tcFor)

tcAlias – Aliasname, der für die Berichtsausgabe verwendet wird.

tcRezFile – Vollständiger Pfadname der zu erstellenden PDF-Datei.

tcFRXName – Name der Berichtsdatei, die zur Erstellung der PDF-Datei verwendet wird.

tcFor – For-Klausel zur Filterung der zu exportierenden Daten.

Diese Methode gibt den Wert .T. zurück, wenn die PDF-Datei erfolgreich erstellt werden konnte. .F. wird zurückgegeben, wenn die PDF-Datei nicht erstellt werden konnte. In diesem Fall sind die Nummer und die Beschreibung des aufgetretenen Fehlers in den Eigenschaften *LastErrorNo* und *LastErrorText* gespeichert.

11.33. Die Klasse cEmail

Diese Klasse gibt dem Entwickler die Möglichkeit E-Mails zu versenden. Es müssen nur wenige Parameter der Methode *Send_Email_Report* übergeben werden um eine Berichtsausgabe im PDF-Format als E-Mail-Anhang versenden zu können.

11.33.1. Eigenschaften

LastErrorNo – Diese Eigenschaft enthält die Nummer des letzten Fehlers (falls ein Fehler aufgetreten ist). Damit kann die Ursache des letzten Fehlers ermittelt werden.

LastErrorTest – Wenn ein Fehler aufgetreten ist, ist in dieser Eigenschaft der Text der Fehlermeldung zu finden.

oEmail_Attachment – Diese Eigenschaft wird nur intern verwendet. Sie enthält eine Collection der Anhänge.

11.33.2. Methoden

AddAttachment (tsAlias, tcFileName, tcReport, tcFor)

Fügt dem E-Mail-Objekt Informationen über einen E-Mail-Anhang hinzu, der mit der nächsten E-Mail gesendet wird. Die Informationen über alle vorzubereitenden PDF-Anhänge werden in der Eigenschaft *oEmail_Attachment* gespeichert. Wenn der Aliasname einer geöffneten Tabelle oder Ansicht angegeben und der Name einer Berichtsdatei

übergeben wird, wird diese Klasse automatisch eine PDF-Datei zu dem Bericht erstellen. Es kann ein weiterer Ausdruck als Parameter angegeben werden, der dazu verwendet wird die Daten des Berichts zu filtern. Wenn kein Aliasname angegeben wird und keine Tabelle im aktuellen Arbeitsbereich geöffnet ist, nimmt die Klasse an, dass ein Dateianhang vorbereitet wurde. In diesem Fall muss die Datei existieren, wenn die Methode *Send_Email_Report* aufgerufen wird.

tcAlias – Aliasname, der für die Berichtsausgabe und für den PDF-Export verwendet wird.

tcRezFile – Name des Dateianhangs (wenn eine PDF-Datei erstellt wird, wird dies der Name der PDF-Datei).

tcFRXName – Name der Berichtsdatei, aus der die PDF-Datei erstellt wird.

tcFor – For-Klausel mit der die Berichtsdaten für die PDF-Ausgabe gefiltert werden.

Send_Email_Report (tcEmail, tcSubject, tcText)

Sendet eine E-Mail. Wenn die E-Mail mit Anhängen versendet werden soll, müssen diese vorher mit der Methode *AddAttachment* angefügt werden.

tcEmail – Adresse des E-Mail-Empfängers.

tcSubject – Betreff der E-Mail.

tcText – Text der E-Mail.

ClearAttachment

Löscht alle E-Mail-Anhänge.

Die Methode *AddAttachment* kann entsprechend der Anzahl der benötigten Anhänge beliebig oft aufgerufen werden. Es werden die Aliasnamen der Tabellen oder Ansichten, die Namen der zu erstellenden Dateien, die Namen der Berichtsdateien und eventuell zu verwendende For-Klauseln als Parameter übergeben. Dann wird die Methode *Send_Email_Reports* aufgerufen. Alle PDF-Dateien werden erstellt und als E-Mail-Anhänge versendet. Auch die Dateien, die zuvor vorbereitet wurden und als Anhang versendet werden sollen, werden an die E-Mail angehängt.

11.34. Die Klasse cArchive

Diese Klasse dient der Datensicherung und Datenwiederherstellung. Die Daten werden in Zip-Archiven gesichert. Der Name des Archivs wird aus dem Namen des Datenordners und dem aktuellen Datum in ANSI-Form zusammengesetzt. Wenn zum Beispiel der Datenordner „Data“ heißt und die Datensicherung am 4. November 2003 durchgeführt wird, heißt das Archiv Data20031104.zip.

11.34.1. Eigenschaften

OverrideFile – Mit dieser Eigenschaft wird festgelegt was passiert, wenn eine Datei mit dem gleichen Namen schon vorhanden ist.

0 – Vorgang abbrechen, wenn bereits eine Datei mit dem gleichen Namen existiert.

1 – Wenn eine Datensicherung durchgeführt wird, werden neue Dateien dem Archiv hinzugefügt und bestehende Dateien werden aktualisiert. Wenn eine Wiederherstellung durchgeführt wird, werden existierende Dateien nicht überschrieben.

2 – Wenn eine Datensicherung durchgeführt wird, wird ein bestehendes Archiv überschrieben. Wenn eine Wiederherstellung durchgeführt wird, werden existierende Dateien überschrieben.

OperationSuccessfully – Enthält das Ergebnis der letzten Aktion.

.T. – wenn die Aktion erfolgreich ausgeführt werden konnte.

.F. – wenn die Aktion nicht ausgeführt werden konnte.

11.34.2. Methoden

CreateArchive (lcFileLocation, lcMask, lcArchFilePathName)

lcFileLocation – Vollständiger Pfad zu dem Ordner, dessen Inhalt gesichert werden soll.

lcMask – Zu sichernde Dateien, Beispiel: „*.DBF;*.FPT;*.CDX“.

lcArchFilePathName – Vollständiger Pfadname der zu erstellenden Archivdatei.

Rückgabewert: .T. – wenn die Aktion erfolgreich ausgeführt werden konnte, .F. – wenn die Aktion nicht ausgeführt werden konnte.

ZipProgress (*tcCurrentOperatedFile, nState, nAllFilesSize, nZIPedFilesSize, nArchive-CurrentSize)* Callback-Funktion der *CreateZipArchive*-Funktion (in VFX.fll).

tcCurrentOperatedFile – Der Name der Datei, die dem Archiv hinzugefügt wird.

nState – Aktuelle Aktion:

1 – Datei existiert

2 – Datei wird dem Archiv hinzugefügt

3 – Datei erfolgreich dem Archiv hinzugefügt

4 – Datei konnte dem Archiv nicht hinzugefügt werden

5 – Archivierungsvorgang erfolgreich beendet

6 – Archivierungsvorgang nicht erfolgreich beendet

7 – Keine Dateien zu archivieren

nAllFilesSize – Die Größe aller zu archivierenden Dateien.

nZIPedFilesSize – Die Größe der dem Archiv bereits hinzugefügten Dateien.

nArchiveCurrentSize – Die aktuelle Größe des Archivs.

Rückgabewert: 0 – Abbruch der Aktion. 1 – Fortsetzen Dateien dem Archiv hinzuzufügen und existierende Dateien zu überschreiben. 2 – Bestehende Archivdatei überschreiben

ExtractFromArchive(lcArchFileForExtract, lcPathForExtract)

lcArchFileForExtract – Vollständiger Pfadname der zu entpackenden Zip-Datei.

lcPathForExtract – Zielordner, in den die Dateien entpackt werden sollen.

UnZipProgress (tcCurrentOperatedFile, nState, nArchiveFilesSize, nUnZIPedFilesSize) Callback-Funktion der ExtractZipArchive-Funktion (in VFX.fll).

tcCurrentOperatedFile – Name der aktuell entpackten Datei aus dem Archiv.

nState – Aktuelle Aktion

1 – Datei existiert bereits

2 – Datei wird entpackt

3 – Datei entpacken beendet

4 – Datei konnte nicht entpackt werden

5 – Entpacken des Archiv erfolgreich abgeschlossen

6 – Entpacken des Archiv nicht erfolgreich abgeschlossen

nArchiveFilesSize – Größe des Archivs

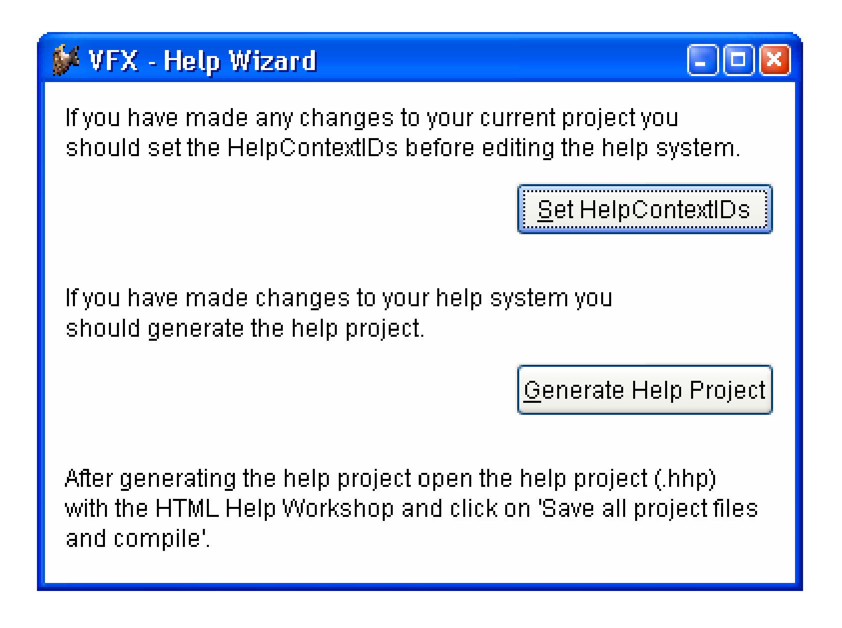
nUnZIPedFilesSize – Größe des Teils des Archivs, das bereits entpackt wurde.

Rückgabewert: 0 – Abbruch der Aktion. 1 – Aktuelle Datei nicht entpacken.2 – Vorhandene Datei überschreiben.

11.35. VFX – Help Wizard

In VFX ist ein System zur Erstellung von CHM-Hilfedateien integriert.

Der VFX – Help Wizard trägt in alle Steuerelemente eines Projekts automatisch eindeutige *HelpContextIDs* ein.



Wenn zur Laufzeit der Anwendung die Tabelle *Vfxhelp.dbf* zur Verfügung steht, können Hilfetexte in diese Tabelle erfasst werden. Dafür wird das Formular *Vfxhelp.scx* geöffnet. Der eigentliche Hilfetext wird in einer Editbox erfasst und in der Tabelle *Vfxhelp.dbf* gespeichert.

Mittels des VFX – Help Wizard können aus den Daten der Tabelle *Vfxhelp.dbf* vollautomatisch HTM-Dateien sowie ein Hilfe-Projekt erstellt werden. Mit dem Help-Workshop muss dieses Projekt nur noch kompiliert werden und die CHM-Hilfedatei mit kontextsensitiver Hilfe zur gesamten Anwendung ist fertig.

Wenn die Tabelle *Vfxhelp.dbf* zur Laufzeit der Anwendung nicht zur Verfügung steht, wird das normale kontextsensitive Hilfesystem aktiviert. Die CHM-Hilfedatei wird geöffnet und als Parameter wird die *HelpContextID* des aktiven Steuerelements übergeben.

11.36. Die Weiterentwicklung mit VFP

Das gesamte VFX 8.0-Projekt liegt in normalen VFP Quelldateien vor. Die erstellte Anwendung kann also jederzeit mit VFP weiterentwickelt werden, auch wenn auf dem Entwicklungsrechner VFX nicht installiert ist.

12. VFX.fll

Die Datei *VFX.fll* enthält zahlreiche Funktionen, die für die Produktaktivierung, die Datensicherung sowie für den Zugriff auf das Internet benötigt werden. Die *VFX.fll* muss zusammen mit den Anwendungen an die Kunden ausgeliefert werden. Die Funktionen der *VFX.fll* werden im Einzelnen beschrieben.

12.1. Internet, E-Mail und Hilfsfunktionen

URLDownload2File(cUrl, cFileName, cFeedBackFunction, cCancelDownload) – Download einer Datei aus dem Internet. Ein Beispiel für die Anwendung dieser Funktion befindet sich in der Klasse *cDownload* in der Methode *download*.

cUrl – URL der Datei, die heruntergeladen werden soll.

cFileName – Datei- oder Pfadname. Hier wird die heruntergeladene Datei gespeichert.

cFeedBackFunction – Name einer Funktion oder Methode, die von *URLDownload2File* aufgerufen wird, um Informationen über den Fortschritt zu liefern. Die Funktion oder Methode muss zwei Parameter akzeptieren.

cFeedBackFunction(nCurrentAmount, nFileSize)

nCurrentAmount – Anzahl der bereits heruntergeladenen Bytes.

nFileSize – Größe der herunterzuladenden Datei.

cCancelDownload – Name einer Variablen oder Eigenschaft, die den Fortgang des Downloads steuert. Die Variable oder Eigenschaft wird automatisch ständig überprüft.

cCancelDownload = *.F.* – Der Download wird fortgesetzt.

cCancelDownload = *.T.* – Der Download wird abgebrochen.

Rückgabewert: 0 – Der Download wurde erfolgreich abgeschlossen.

Get_PS_Printers(nLocation, @cPrinterNames, @nPrinterNamesLength) – Liefert die Namen aller installierten Postscript-Druckertreiber. Ein Beispiel für die An-

wendung dieser Funktion befindet sich in der Klasse *cCreatePDF* in der Methode *checkpsprinter*.

nLocation – Standort des Druckers.

1 – Es wird nach lokalen Druckern gesucht.

2 – Es wird nach Netzwerkdruckern gesucht.

3 – Es wird lokalen Druckern und Netzwerkdruckern gesucht.

cPrinterNames – Enthält die Namen aller installierten Postscript-Druckertreiber in einer Komma-separierten Liste.

nPrinterNamesLength – Länge der zurückgegebenen Zeichenkette.

Rückgabewert: 0 – Der Vorgang wurde erfolgreich ausgeführt.

Add_Printer(cPrinterName ,cPrinterPort) – Vollautomatische Installation eines Druckertreibers. Ein Beispiel für die Anwendung dieser Funktion befindet sich in der Klasse *cCreatePDF* in der Methode *checkpsprinter*.

cPrinterName – Name des zu installierenden Druckertreibers.

cPrinterPort – Anschluss des zu installierenden Druckertreibers.

Rückgabewert: 0 – Die Installation wurde erfolgreich abgeschlossen.

Encrypt(cStringForEncrypting, cPassword) – Verschlüsselung einer Zeichenkette mit einem Kennwort. Ein Beispiel für die Anwendung dieser Funktion befindet sich in der Klasse *cDunConnectiom.cmdOk* in der Methode *click*.

cStringForEncripting – Zu verschlüsselnde Zeichenkette.

cPassword – Das zur Verschlüsselung dienende Kennwort.

Rückgabewert: Verschlüsselte Zeichenkette.

Decrypt(cStringForDecripting ,cPassword) – Entschlüsselung einer Zeichenkette mit einem Kennwort. Ein Beispiel für die Anwendung dieser Funktion befindet sich in der Klasse *cDunConnection* in der Methode *init*.

cStringForDecripting – Zu entschlüsselnde Zeichenkette.

cPassword – Das zur Entschlüsselung dienende Kennwort.

Rückgabewert: Entschlüsselte Zeichenkette.

GetAxControlSize(nhWnd, @nWidth, @nHeight) – Rückgabe der Größe eines ActiveX-Steuerelements. Ein Beispiel für die Anwendung dieser Funktion befindet sich in der Klasse *cCalendar* in der Methode *resize*.

nhWnd – Handle des Fensters des ActiveX-Steuerelements.

nWidth – Breite des ActiveX-Steuerelements.

nHeight – Höhe des ActiveX-Steuerelements.

Rückgabewerte:

.T. – Die Größe des ActiveX-Steuerelements konnte erfolgreich ermittelt werden.

.F. – Die Größe des ActiveX-Steuerelements konnte nicht ermittelt werden.

SetModemConnection(cConnectionName, cPhoneNumber, cUserName, cPassword) – Einrichten einer DFÜ-Netzwerkverbindung. Ein Beispiel für die Anwendung dieser Funktion befindet sich in der Klasse *cDownload* in der Methode *establishdunconnection*. Für die erfolgreiche Ausführung dieser Funktion muss ein Modemtreiber installiert sein!

cConnectionName – Name der zu erstellenden DFÜ-Netzwerkverbindung.

cPhoneNumber – Zu wählende Rufnummer.

cUserName – Benutzername der Verbindung.

cPassword – Kennwort der Verbindung.

Rückgabewert:

.T. – Die DFÜ-Netzwerkverbindung wurde erfolgreich angelegt.

.F. – Die DFÜ-Netzwerkverbindung konnte nicht angelegt werden.

CheckInetConn(cCheckURL, cDUNConnName, nHWnd) – Diese Funktion überprüft, ob eine Verbindung mit dem Internet besteht. Hierzu wird eine URL im Internet aufgerufen. Ein Beispiel für die Anwendung dieser Funktion befindet sich in der Klasse *cDownload* in der Methode *checkinternetconnection*.

cCheckURL – Diese URL wird überprüft um festzustellen, ob eine Verbindung mit dem Internet besteht.

cDUNConnName – Über diese DFÜ-Netzwerkverbindung wird bei Bedarf eine Verbindung hergestellt.

nHWnd – Handle des aufrufenden Fensters.

Rückgabewerte:

0 – Es besteht eine Verbindung mit dem Internet.

1 – Die Verbindungsherstellung wurde durch den Benutzer abgebrochen.

2 – Es besteht keine Verbindung mit dem Internet.

3 – Es ist ein Fehler aufgetreten.

24 – Die DFÜ-Netzwerkverbindung mit dem Namen *cDUNConnName* existiert nicht.

12.2. Produktaktivierung

GetAppRights(lcRightsBin, This.Hex2Bin(This.cActPattern)) – Liefert Informationen über ein Recht aus der Produktaktivierung. Ein Beispiel für die Anwendung dieser Funktion befindet sich in der Klasse *cVFXActivate* in der Methode *checkactstate*.

Rückgabewert:

0 – Der Vorgang wurde erfolgreich ausgeführt.

-1 – Die Länge des Aktivierungsschlüssels ist ungültig.

-2 – Der Aktivierungsschlüssel ist inkonsistent.

-3 – Fehler bei der Verschlüsselung.

GetFileCreationDateTime(cFileName) – Liefert Datum und die Uhrzeit zu der eine Datei erstellt wurde. Ein Beispiel für die Anwendung dieser Funktion befindet sich in der Klasse *cVFXActivate* in der Methode *init*.

cFileName – Name der zu überprüfenden Datei.

Rückgabewert: Ein Zeit/Datum-Wert als Zeichenkette.

GetSysInfo(This.Hex2bin(This.cActPattern)) – Diese Funktion liefert den Installationsschlüssel. Ein Beispiel für die Anwendung dieser Funktion befindet sich in der Klasse *cVFXActivate* in der Methode *checkactstate*.

12.3. Datensicherung oder Archivierung

CreateZipArchive(cPath, cFileMask, cArchiveFullPathName, cFeedBackFunction) – Erstellen einer Zip-Archivdatei. Ein Beispiel für die Anwendung dieser Funktion befindet sich in der Klasse *cArchive* in der Methode *createarchive*.

cPath – Pfad des zu archivierenden Ordners.

cFileMask – Namen der zu archivierenden Dateien.

cArchiveFullPathName – Pfad- und Dateiname des zu erstellenden Zip-Archivs.

cFeedBackFunction – Name einer Funktion oder Methode, die von *CreateZipArchive* aufgerufen wird und Informationen über den Fortschritt zu liefern.

cFeedBackFunction(cCurrentOperatedFile, nState, nAllFilesSize, nZIPedFilesSize, nArchiveCurrentSize) – Diese Funktion oder Methode wird von *CreateZipArchive* immer dann aufgerufen

wenn die zu erstellende Zip-Datei bereits existiert,

bevor eine Datei dem Archiv hinzugefügt wird,

nachdem eine Datei dem Archiv hinzugefügt wurde,

nachdem ein Archiv erfolgreich erstellt wurde,

wenn ein Archiv nicht erstellt werden konnte,

eine Datei nicht dem Archiv hinzugefügt werden konnte.

cCurrentOperatedFile – Name der Datei, die zurzeit bearbeitet wird.

nState – Status.

1 – Die Datei *cArchiveFullPathName* existiert bereits.

2 – Beginn des Hinzufügens der Datei *cCurrentOperatedFile* zum Archiv.

3 – Ende des Hinzufügens der Datei *cCurrentOperatedFile* zum Archiv.

4 – Die Datei *cCurrentOperatedFile* konnte dem Archiv nicht hinzugefügt werden.

5 – Die Erstellung des Archiv wurde vollständig abgeschlossen.

6– Die Erstellung des Archivs konnte nicht abgeschlossen werden.

7 – Es wurde kein gültiger Pfad- oder Dateiname angegeben bzw. es sind keine Dateien zu archivieren.

nAllFilesSize – Gesamtgröße aller Dateien, die dem Archiv hinzugefügt werden sollen.

nZIPedFilesSize – Größe der Dateien, die dem Archiv bereits hinzugefügt wurden.

nArchiveCurrentSize – Momentane Größe der erstellten Archivdatei.

Rückgabewert:

0 – Der Vorgang wurde abgebrochen.

1 – Die Dateien wurden dem Archiv hinzugefügt.

2 – Der Vorgang wird fortgesetzt.

ExtractZipArchive(cExtractFilesFolder, cArchiveFullPathName, cFeedBackFunction) Entpacken von Dateien aus einer Zip-Archivdatei. Ein Beispiel für die Anwendung dieser Funktion befindet sich in der Klasse *cArchive* in der Methode *extractfromarchive*.

cExtractFilesFolder – Ordner, in den die entpackten Dateien gespeichert werden.

cArchiveFullPathName - Name und Pfadname der Archivdatei.

cFeedBackFunction – Name einer Funktion oder Methode, die aufgerufen wird um Informationen über den Fortschritt zu liefern.

cFeedBackFunction(cCurrentOperatedFile, nState, nArchiveFilesSize, nUnZIPedFilesSize) – Diese Funktion oder Methode wird von *cFeedBackFunction* immer dann aufgerufen wenn

eine zu entpackende Datei bereits existiert,

das Entpacken einer Datei beginnt,

das Entpacken einer Datei endet,

eine Datei nicht aus dem Archiv entpackt werden kann,

das Entpacken aller Dateien erfolgreich abgeschlossen wurde,

das Entpacken aller Dateien nicht abgeschlossen werden konnte.

cCurrentOperatedFile – Name der zurzeit entpackten Datei.

nState Status

1 – Die zurzeit bearbeitete Datei existiert bereits.

2 – Beginn des Entpackens der Datei *cCurrentOperatedFile*.

3 – Ende des Entpackens der Datei *cCurrentOperatedFile*.

4 – Die Datei *cCurrentOperatedFile* konnte nicht entpackt werden.

5 – Der Vorgang wurde erfolgreich abgeschlossen.

6 – Der Vorgang konnte nicht abgeschlossen werden.

Rückgabewert:

0 – Abbruch des Entpackens.

1 – Fortsetzen des Vorgangs.

2 – Überschreiben der bestehenden Datei mit der Archivdatei.

12.4. SQL Server

GetSQLServers(@cServersString, @cEerrorString) – Ermitteln aller verfügbaren SQL Server. Ein Beispiel für die Anwendung dieser Funktion befindet sich in der Funktion *TryConnecting* in *Vfxfunc.prg*.

cServersString – Zeichenkette, die eine durch Komma getrennte Liste mit den Namen aller verfügbaren SQL Server enthält.

cErrorString – Eventuell aufgetretene Fehler werden hier zurückgegeben.

Rückgabewert: Anzahl der ermittelten SQL Server.

GetSQLDataBases(cServer, @cDBString, cUser, cPass, @cErrors) – Ermitteln aller Datenbanken eines SQL Servers.

cServer – Name des SQL Servers von dem die Datenbanken ermittelt werden sollen.

cDBString – Eine Zeichenkette mit den durch Komma getrennten Namen aller verfügbaren Datenbanken.

cUser – Benutzername für die Anmeldung beim SQL Server.

cPass – Kennwort für die Anmeldung beim SQL Server.

cErrors – Eventuelle Fehlermeldung des SQL Servers.

Rückgabewert: *0* – Der Vorgang wurde erfolgreich abgeschlossen.

13. Erstellen mehrsprachiger Anwendungen mit VFX

VFX ist gut vorbereitet, um mehrsprachige Anwendungen zu erstellen. Sie können zwischen Lokalisierung zur Laufzeit und Lokalisierung während der Entwicklung wählen. Hier beschreiben wir den Vorgang der Lokalisierung während der Entwicklung.

Die Bedienungselemente tauchen in den folgenden Bereichen auf:

- Bedienung der bestehenden Funktionalität in den Visual Extend-Klassenbibliotheken und allen Dialogen.
- Bedienung Ihrer eigenen Anwendung.

Sie brauchen sich nicht um die ersten beiden Punkte zu kümmern.

Die Bedienungselemente der bestehenden Funktionalität in den Visual Extend-Klassenbibliotheken und allen Dialogen existieren in zurzeit sieben Sprachen. Sie brauchen kein Wort zu übersetzen, wenn Ihre Anwendung auf *deutsch, englisch, französisch, italienisch, griechisch, bulgarisch, tschechisch* oder *spanisch* erstellt werden soll. Wenn Sie die Visual Extend-Klassenbibliotheken in einer anderen Sprache benötigen, können Sie VFX leicht selbst erweitern.

Wir wären Ihnen sehr dankbar, wenn Sie uns Ihre Übersetzung der VFX-Meldungen in der Tabelle *VFXMSG.DBF/CDX/FPT* in eine andere Sprache als *deutsch, englisch, französisch, italienisch, griechisch, bulgarisch, tschechisch* und *spanisch* zusenden würden. Wir könnten diese dann anderen Entwicklern zur Verfügung stellen. Vielen Dank!

Prüfliste für die Erstellung mehrsprachiger Anwendungen mit VFX:

√ Benutzen Sie die Include-Dateien *USERTXT.H* bzw. *USERMSG.H*, die vom **VFX – Message Editor** erstellt werden, um alle sprachabhängigen Bedienungselemente für Ihre Anwendung zu verwalten. ***Der Speicher für Bezeichnungen, Meldungen, Überschriften, Tooltip-Texte und Statuszeilenmeldungen ist die Tabelle VFXMSG.DBF. In dieser Tabelle finden Sie auch alle von VFX benutzten Texte, die bereits in die zur Verfügung stehenden Sprachen übersetzt sind.***

√ Benutzen Sie in Ihrer Anwendung Konstanten anstelle von direkten Texten, z. B. **WAIT WINDOW Loc_Text1** anstelle von **WAIT WINDOW "MyText…".**

√ Benutzen Sie die Include-Datei *USERDEF.TXT* für alle anwendungsspezifischen Konstanten, die sprachunabhängig sind. Dadurch wird Ihre Lokalisierungsarbeit erleichtert.

√ Benutzen Sie den **VFX – LangSetup Builder,** um den Code für die VFX-Formularmethode mit dem Namen *LangSetup()* zu erstellen. Die Methode enthält den Lokalisierungscode. Den Bezeichnungen, Tooltip-Texten usw. werden die Werte aus den Konstanten zugewiesen. (Der VFX – LangSetup Builder erzeugt automatisch den Code für die *LangSetup()*-Methode und aktualisiert die Tabelle *VFXMSG.DBF* mit den Meldungen und Bezeichnungen.)

√ Übersetzen Sie Ihren Text mit dem ***VFX – Message Editor*** in die verschiedenen Sprachen. Der VFX-Message-Editor erzeugt Include-Dateien für die verschiedenen Sprachen im Ordner *INCLUDE**LanguageDir*. *LanguageDir* steht für den Namen der Sprache, in die Sie übersetzen. (Wie oben bereits erwähnt, wurden die VFX-spezifischen Sprachkonstanten bereits in einige Sprachen übersetzt. Sie brauchen hierfür kein einziges Wort mehr zu übersetzen.)

√ Um Ihre Anwendung für eine Sprache zu erstellen, definieren Sie die Konstante *ID_LANGUAGE* in der *VFXDEF.H*-Include-Datei und kopieren Sie die Include-Datei aus dem Ordner *INCLUDE**LanguageDir* in den aktuellen *INCLUDE*-Ordner Ihres Projektes.

√ Wählen Sie die Option *Alle Dateien nwu kompilieren* und testen Sie Ihre Anwendung. Sie erhalten für jede Sprache eine eigene EXE-Datei.

14. Portierung einer Anwendung von VFX 7 nach VFX 8.0

Bestehende VFX 7-Anwendungen können einfach nach VFX 8.0 portiert werden. Da sich zwischen VFX 7 und VFX 8.0 vieles geändert hat, ist es empfehlenswert zunächst ein neues, leeres Projekt mit VFX 8.0 zu erstellen. Dieses neue Projekt wird in einem neuen Verzeichnis erstellt, bekommt aber den Namen und insbesondere auch den Datenbanknamen des alten Projekts. Alle Eigenschaften im VFX – Application Wizard werden wie beim alten VFX 7 Projekt eingetragen.

Im neuen VFX 8.0-Projekt stehen damit automatisch die neuen Menüs von VFX 8.0, die erweiterten Include-Dateien sowie die erweiterten Strukturen der freien Tabellen zur Verfügung. Wenn Sie das Menü *Vfxmenu* Ihres alten Projektes erweitert hatten, machen Sie die Erweiterungen in dem neuen Menü bitte manuell. Nur so bleiben die erweiterten Menüeinträge von VFX 8.0 erhalten.

Prüfen Sie das neue *VFXMAIN.PRG* und machen Sie von Hand die für Ihr Projekt erforderlichen Änderungen. Folgen Sie der Dokumentation im neuen *VFXMAIN.PRG*, um die Vorlage an Ihre spezifischen Bedürfnisse anzupassen.

Aus dem VFX 7-Projekt kann jetzt die Datenbank in den Data-Ordner des neuen Projekts kopiert werden. Die freien VFX-Tabellen werden im VFX 8.0-Projekt mit *USE* geöffnet und mit *APPEND FROM* werden die Daten aus dem alten VFX 7-Projekt der jeweiligen Tabelle angefügt. Auf diese Weise müssen die Daten der Tabellen *Vfxfopen.dbf*, *Vfxsys.dbf*, *Vfxsysid.dbf* und *Vfxusr.dbf* geholt werden.

Alle Formulare und Berichte werden aus den jeweiligen Ordnern, Form bzw. Report, in das neue Projekt kopiert und manuell dem VFP Projekt-Manager hinzugefügt. Schließlich wird noch die Datei *Applfunc.prg* aus dem *Program*-Ordner und die Dateien *Appl.vc** aus dem *Lib*-Ordner in das neue Projekt kopiert.

Wenn Sie eigene Konstanten in Include-Dateien abgelegt haben, kopieren Sie Ihre Include-Dateien in den *Include*-Ordner des neuen Projektes. Überschreiben Sie nicht die Include-Dateien *Vfxmsg.h* und *Vfxtxt.h*, da diese zahlreiche neue Konstanten enthalten, die von VFX 8.0 benötigt werden.

Nach dem Kompilieren aller Dateien ist die Anwendung mit VFX 8.0 einsatzbereit.

15. Dokumentation

Neben dem Benutzerhandbuch gibt es zu VFX eine Menge an Online-Dokumentation. Dazu gehört insbesondere die Technische Referenz, die als Windows-Hilfedatei vorliegt. In ihr ist zu jeder Klassenbibliothek, zu jeder Klasse jede Methode und jede Eigenschaft beschrieben. In einem Tutorial werden anhand von typischen Anwenderfragen die Lösungen mit VFX erläutert. Direkt aus der Technischen Referenz können Videos (Avi-Dateien) gestartet werden. Es gibt 10 Videos mit insgesamt ca. 45 Minuten Dauer. In den Videos wird die Erstellung von Formularen für Fileserver- und Client-/Server-Datenbanken beschrieben und gezeigt. Für den VFX-Anfänger eine große Hilfe bei der Einarbeitung.

15.1. Support

Support für VFX ist im dFPUG-Forum (http://forum.dfpug.de) zu finden. Dort gibt es sowohl eine deutsche als auch eine englische Sektion zu VFX. Diese Sektionen können auch alternativ als Newsgroup (news://news.dfpug.de) gelesen und bearbeitet werden.

Im Internet findet man auf der Website von Visual Extend (http://www.visualextend.de) weitere Informationen zum Produkt. Auch ist hier der Download der Demoapplikation, der gesamten Dokumentation und der aktuellen Vollversion von VFX möglich. Eine umfangreiche Sammlung weiterer Dokumente rund um VFX findet sich im Dokumentenportal der dFPUG (http://portal.dfpug.de). Aktuelle Informationen erhalten Sie über den kostenlosen dFPUG-eNewsletter im Abschnitt zu VFX (http://newsletter.dfpug.de).

16. Zusammenfassung

Wie wir gesehen haben stellt VFX eine vollständige Entwicklungsumgebung bereit, die keine Wünsche offen lässt. Alle wesentlichen Einstellungen an VFX-Klassen, insbesondere an den Formularklassen, können mit reentranten Buildern durchgeführt werden. Alle in diesem Artikel beschriebenen Eigenschaften und Funktionen lassen sich praktisch ohne Programmierung nur durch den Einsatz der Builder erreichen.

Trotzdem ist es an praktisch jeder Stelle über Hooks möglich in den Programmablauf einzugreifen.

Da VFX mit Quellen geliefert wird und selbst mit VFP programmiert ist, hat der Entwickler unbegrenzte Freiheit eigene Erweiterungen oder Anpassungen an eigene Bedürfnisse vorzunehmen.

Die Performance von VFX-Anwendungen ist so gut, wie sie mit VFP-Anwendungen nur sein kann. Die Vererbungstiefe ist gering. Die meisten Klassen haben nur 1 bis 2, maximal jedoch 4 Vererbungsebenen hinter sich. Um das Laden von umfangreichen Formularen weiter zu beschleunigen kann Delayed Instantiation verwendet werden. Auch dies wird von VFX mit einfach zu handhabenden Funktionen unterstützt.

Die mit VFX erstellten Applikationen vermitteln dem Anwender einen sehr professionellen Eindruck und eine Office-kompatible Bedienung.

VFX bietet mit all dem ein unschlagbares Preis-/Leistungsverhältnis. Es bietet jedem Programmierer eine Fundgrube an Ideen und eine Vielzahl von fertigen Problemlösungen.

16.1. Ihre Meinung ist uns wichtig!

Senden Sie uns Ihre Meinung via eMail an visualextend@dfpug.de oder besuchen Sie unsere VFX Newsgroup unter news://news.dfpug.de.

Wir danken allen VFX-Kunden für das bisherige, großartige Feedback!

VFX 8.0 - Produktiver als je zuvor!

17. Anhang 1 – Erweiterung: Active Desktop Benutzeroberfläche für VFX

17.1. Active Desktop Singleclick-Benutzeroberfläche

17.1.1. Visual Extend

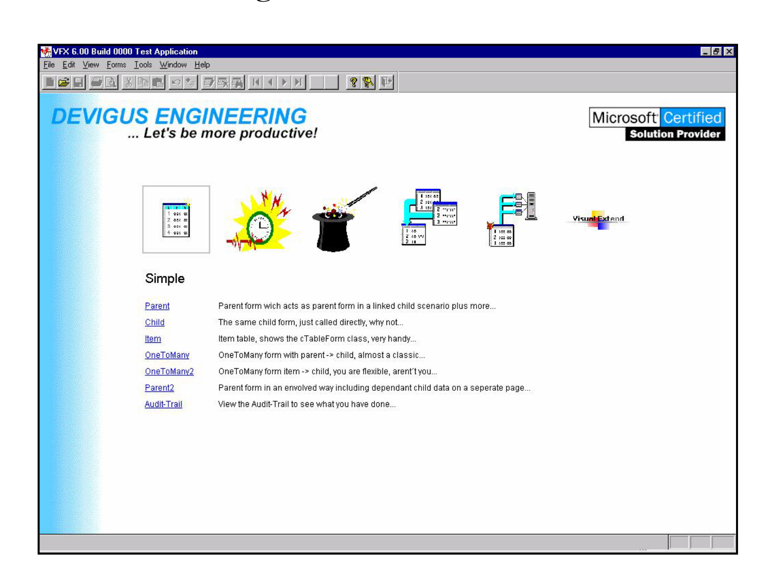
Visual Extend (VFX) ist ein Anwendungsentwicklungs-Framework (RAD) für Softwareentwickler, die mit Microsoft Visual FoxPro arbeiten. Visual Extend beinhaltet Builders die den Entwickler in seiner täglichen Arbeit unterstützt und die Anwendungsentwicklung dramatisch beschleunigt, ohne die Möglichkeiten von Visual FoxPro einzuschränken. Mit Visual Extend wird Visual FoxPro ein richtiges "Rapid Application Development" (RAD) -werkzeug für die Erstellung von Desktop- und Client-Server Datenbankapplikationen.

17.1.2. Active Desktop Klasse

Die Active Desktop Klasse ermöglicht es dem Anwender jede beliebige Funktion einer Anwendung mit einem einzigen Klick zu starten. Diese Klasse „CNavCont" können Sie in der Klassenbibliothek VFXTOOLS.VCX finden und erlaubt es Ihnen eine richtige Ein-Klick-Oberfläche zu entwickeln, wie Sie in der Windowswelt und dem Internet Standard ist.

Zur Nutzung dieser Klasse finden Sie weitere Informationen in der Dokumentation von VFX, sowie dem Worddokument zur Session V-ACTI, beziehbar über das dFPUG-Portal auf http://portal.dfpug.de.

Zur Erinnerung ein Screenshot eines Active Desktop, wie Sie es kennen:



17.1.3. Den Active Desktop erweitern

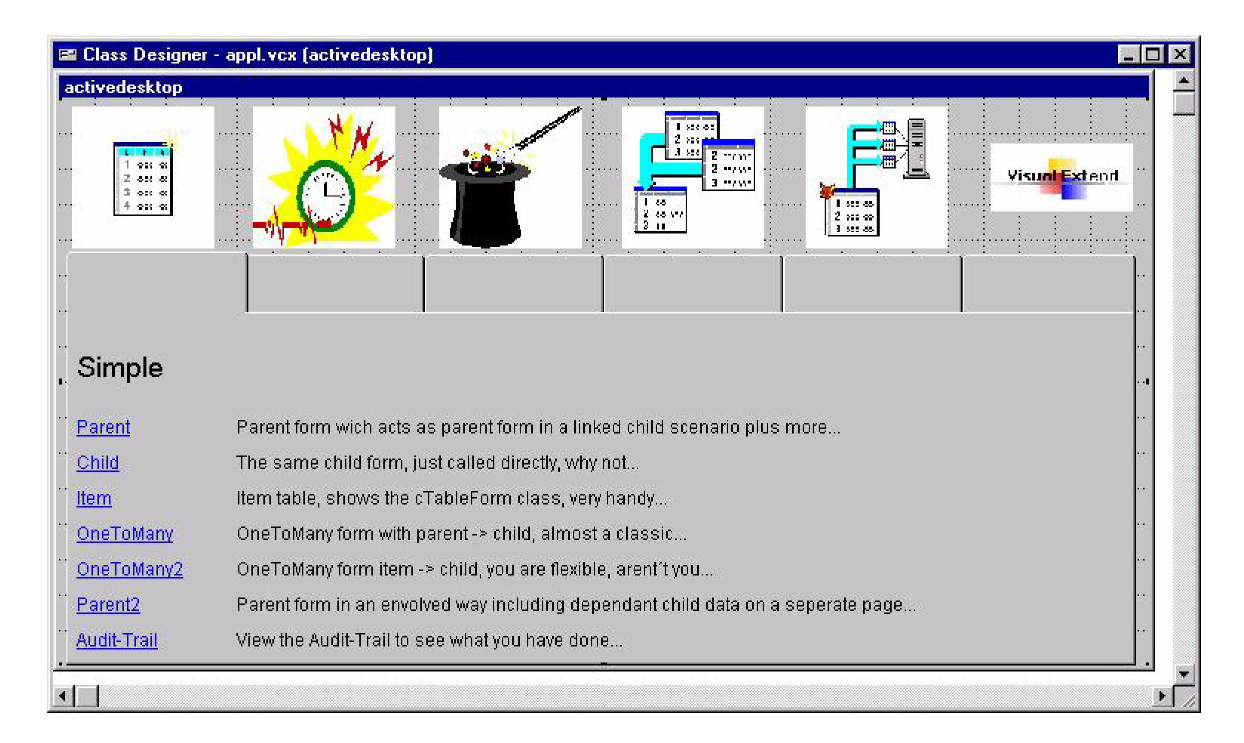
Trotz des schon recht großen vorhandenen Platzes zur Unterbringung von Links auf dem Desktop (sechs Registerkarten), kann es bei größeren Applikationen nötig sein noch weitere Registerkarten zuzufügen, um thematisch alle Funktionen Ihrer Applikation besser unterzubringen.

17.1.4. Wo die Erweiterungen vorgenommen werden

Alle Änderungen und Anpassungen passieren in der applikationseigenen Klassenbibliothek APPL.VCX. Hier ist eine vererbte Klasse CNavCont bei der Erstellung der Applikation angelegt worden. Ändern Sie niemals in der Klassenbibliothek VFXTOOLS.VCX, da diese bei einem Update von VFX überschrieben wird!.

17.1.5. Der Aufbau der Klasse

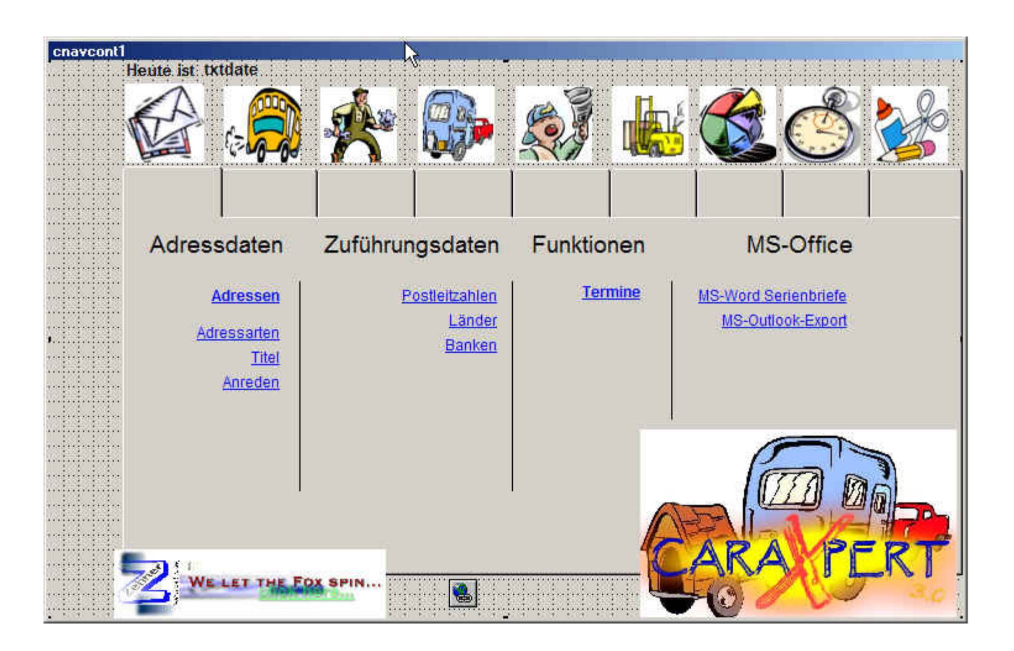
Die Klasse besteht im Wesentlichen auf einem Formular mit Register, sechs Registerseiten, und Grafiken für die Steuerung, wie Sie aus der unteren Abbildung ersehen können.



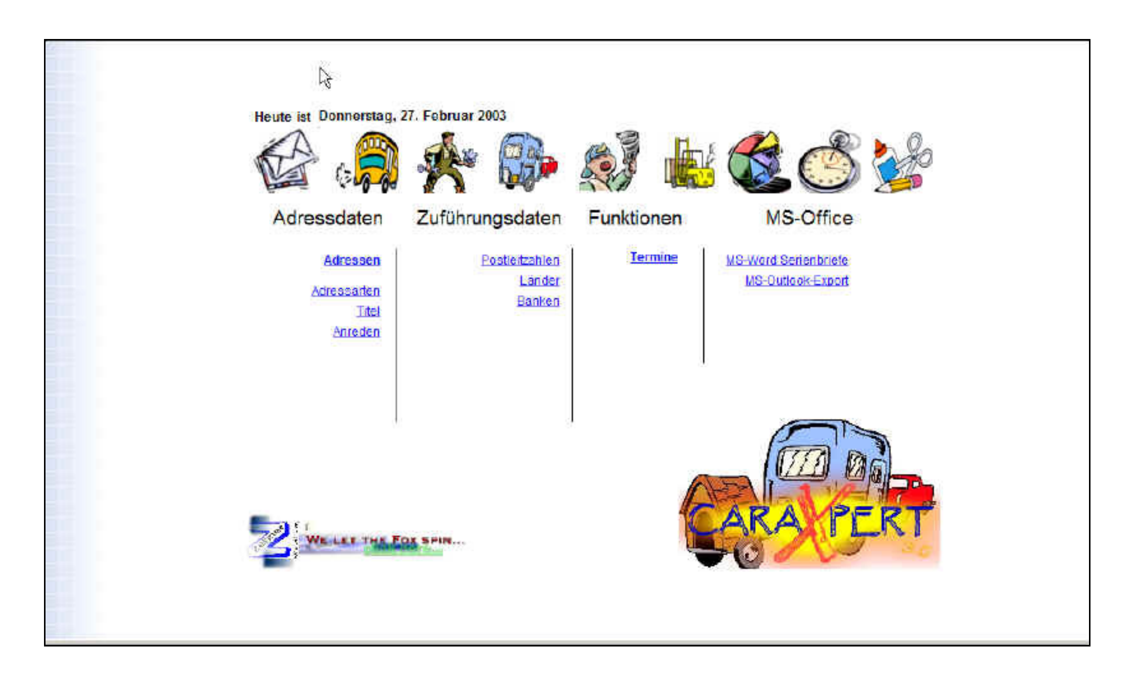
Während der Laufzeit werden die Registerreiter ausgeschaltet, damit man Sie nicht sehen kann. Die weitere Funktionalität des Registers, wie auch der Grafiken ergibt sich aus der Beschreibung der Änderungen.

17.1.6. Das Ziel

Unten sehen Sie ein Beispiel eines erweiterten Desktops. Einmal im Klassendesigner und einmal zur Laufzeit als Desktop. Wie Sie sehen hat dieser Desktop nunmehr 9 Registerkarten. Entsprechend haben Sie also drei weitere Unterteilungsmöglichkeiten für die Funktionen Ihrer Applikation. Ebenso benötigen Sie für eine Erweiterung die entsprechende Anzahl als Grafiken.



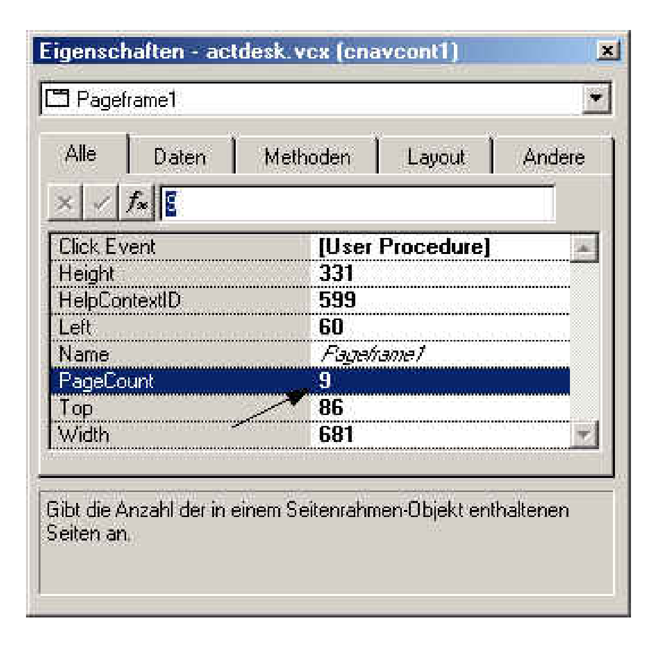
Auf wie viele Registerkarten Sie den Desktop schließlich erweitern ist gänzlich Ihnen überlassen, bedenken Sie aber bitte die evtl. Bildschirmauflösung des Anwenders und die mit der steigenden Anzahl an Grafiken für die Registerkarten geänderten Größenverhältnisse.



17.1.7. Die Änderungen

Nachfolgend erhalten Sie eine detaillierte Beschreibung über die Änderungen, die Sie vornehmen müssen:

1. Ändern Sie auf thisform.pgpgframe.pagecount auf die Anzahl Registerkarten, die Sie wünschen:



2. Stellen Sie die gewünschten zusätzlichen Grafiken im Ordner bitmap Ihrer Anwendung bereit und fügen Sie diese Grafiken Ihrem Projket hinzu.
3. Erzeugen Sie neue Objekte aus der Klasse cbuttonimage, ebenfalls in der Klassenbibliothek VFXTOOLS.VCX vorhanden, in entsprechender Anzahl, je nachdem, wie viele Registerseiten bzw. Grafiken hinzukommen, positionieren und ändern Sie die Größe der Grafikobjekte (width und height).



4. Machen Sie die Code-Anpassungen in den Grafikobjekten, dem Register nebst Karten und der Klasse selbst, wie folgend beschrieben:

17.1.8. Code ändern

Sämtlicher Code, der geändert wird, wird wie o. b. nie in der Parentklasse geändert, sondern in der eigenen abgeleiteten Klasse. Da in der Ableitung natürlich kein Code hinterlegt ist, er wird ja vererbt, können Sie mit DODEFAULT() und zusätzlichem Code arbeiten, wenn keine wirklichen Änderungen des Codes vorgenommen werden müssen. Um persönlich einen besseren Überblick zu behalten, was wo passiert, habe ich mich in diesem Beispiel dazu entschlossen den gesamten Code in meine Ableitungsfunktionen der Klasse zu kopieren und dort direkt anzupassen. Dann rufe ich natürlich auch kein DODEFAULT() auf.

17.1.9. Änderungen in cButtonImage

Durch Einfügen eines neuen ButtonImages auf die Klasse erhält das neue-Buttonimage automatisch den Namen „cButtonImage" gefolgt von einer laufenden Nummer, für den ersten Button also die „7". Sie müssen sich die Nummern bzw. Namen dieser Objekte merken, weil Sie in den Funktionen angesprochen werden. Die Methode oninteractivechange ist schon eingerichtet, es steht aber kein vererbbarer Code zur Verfügung. Diesen müssen Sie sich aus einem anderen ButtonImage kopieren und ändern, oder neu schreiben.

17.1.9.1. OnInterActiveChange des buttons

Hierbei ist zu beachten, dass der Code die Verbindung des Image zu der entsprechenden Registerkarte darstellt. Welche Ziffer also für die Abfrage und die Zuweisung benutzt werden muss. Ist es wie in diesem Falle die 7. Registerkarte, lautet der Code wie folgt:

```
WITH THIS.PARENT
   IF .pageframe1.ActivePage <> 7
      .pageframe1.ActivePage = 7
      .onrefresh()
      THIS.BorderWidth=4
   ENDIF
ENDWITH
```

17.1.9.2. MouseMoveEvent des Buttons

Diese Methode hat den Code aus CButtonImage geerbt und braucht auch nicht geändert werden. Hier wird lediglich die Methode OnInterActiveChange des ButtonImages aufgerufen.

17.1.10. Änderungen in ThisForm

Durch Einfügen eines neuen ButtonImages auf die Klasse erhält das neue-Buttonimage automatisch den Namen „cButtonImage“ gefolgt von einer laufenden Nummer, für den ersten Button also die „7“. Sie müssen sich die Nummern bzw. Namen dieser Objekte merken, weil Sie in den Funktionen angesprochen werden. Die Methode oninteractivechange ist schon eingerichtet, es steht aber kein vererbbarer Code zur Verfügung. Diesen müssen Sie sich aus einem anderen ButtonImage kopieren und ändern, oder neu schreiben.

17.1.10.1. Centerme und OnEventHook

Diese Methoden sind aus CNavCont geerbt und müssen nicht geändert werden.

17.1.10.2. MouseMove

Diese Methode ist aus CNavCont geerbt, muss aber angepasst werden, da hier nur der Code für 6 Registerkarten hinterlegt ist:

```
LPARAMETERS nbutton, nshift, nxcoord, nycoord

WITH THIS
   .centerme()
   .cbuttonimage1.Refresh()
   .cbuttonimage2.Refresh()
   .cbuttonimage3.Refresh()
   .cbuttonimage4.Refresh()
   .cbuttonimage5.Refresh()
   .cbuttonimage6.Refresh()
   .cbuttonimage7.Refresh()    && <- NEU
ENDWITH
```

17.1.10.3. OnRefresh

Diese Methode ist aus CNavCont geerbt, muss aber angepasst werden, da hier nur der Code für 6 Registerkarten hinterlegt ist:

```
WITH THIS
   .cbuttonimage1.BorderWidth=0
   .cbuttonimage2.BorderWidth=0
   .cbuttonimage3.BorderWidth=0
   .cbuttonimage4.BorderWidth=0
   .cbuttonimage5.BorderWidth=0
   .cbuttonimage6.BorderWidth=0
   .cbuttonimage7.BorderWidth=0  && <- NEU
ENDWITH
```

17.1.11. Änderungen in Page1 bis Pagen

Änderungen, bzw. zusätzlicher Code aus jeder Registerkarte betreffen zwei Methoden:

17.1.11.1. MouseMoveEvent

Diese Methode ist aus CNavCont geerbt, muss aber angepasst werden, da hier nur der Code für 6 Registerkarten hinterlegt ist. Zu beachten ist, dass diese Methoden für die 6 vorhandenen Seiten verändert werden müssen und für jede neue Seite ganz neu gefüllt werden müssen, allerdings immer mit dem gleichen Code:

```
LPARAMETERS nbutton, nshift, nxcoord, nycoord

WITH THIS.PARENT.PARENT
   .cbuttonimage1.Refresh()
   .cbuttonimage2. Refresh()
   .cbuttonimage3. Refresh()
   .cbuttonimage4. Refresh()
   .cbuttonimage5. Refresh()
   .cbuttonimage6. Refresh()
   .cbuttonimage7. Refresh()   && <- NEU
   .centerme()
ENDWITH
```

17.1.11.2. Click

Diese Methode ist aus CNavCont geerbet, und muss für geerbte „Pages" nicht geändert werden. Für neue „Pages" muss hier entsprechend den anderen Registerkarten neuer Code rein. Bezogen auf die neue Registerkarte:

```
THIS.PARENT.PARENT.cbuttonimage7.CLICK()
```

17.1.12. Zusammenfassung

Sie sehen, mit ein wenig zusätzlichem Code und minimalem Zeitaufwand können Sie den Active Desktop von VFX erweitern und anpassen. Durch weitere Grafische Elemente, wie Sie sie in meinem Beispieldesktop sehen können, können Sie weitere auflockernde, erklärende und/oder funktionelle Objekte auf den Desktop bringen. Texte können Sie einfach auf die Klassenobjekte verteilen und bei Grafiken, die nicht in cButtonImage eingebettet sind, muss dann lediglich die Prozedur DrawBackground in der Programmdatei vfxmain.prg angepasst werden. Hier nochmals der Verweis zum Worddokument zur Session V-ACTI.

Wie immer garantiere ich für nichts! Sicherlich gibt es in Details andere Möglichkeiten des Active Desktop von VFX zu ändern, scheuen Sie sich nicht Ihre Lösung niederzuschreiben. Sie können auch gerne Hinweise, weitere Verbessungsvorschläge und Anregungen zusenden. Das Gleiche gilt natürlich auch für Fehler, die Ich hier natürlich absichtlich eingebaut habe <g>. Besuchen Sie die Internetseite www.my-vfx.de für weitere Informationen und Kontaktmöglichkeiten.

Probieren Sie diese Anpassungen erst an einem Beispielprojekt aus, bis Sie mit den Vererbungen, Codes und Verhaltensweisen des Active Desktop vertraut sind. Nehmen Sie erst dann Änderungen für Ihr richtiges Projekt vor. Wenn Sie mit dem Active Desktop vertraut sind, dann ist der Arbeitsaufwand es bei einem Anderen Projekt zu wiederholen sehr gering und rechtfertigt dann nicht einmal mehr 7 Seiten Dokumentation.

18. Anhang 2 - Tipps & Tricks mit VFX

Tipps & Tricks mit VFX - Eine Sammlung von Nutzungsmöglichkeiten verschiedener Eigenschaften in VFX und/oder „blankem" VFP.

18.1. Übersicht

Die hier behandelten Themen werden folgend mit Stichwort zur Erklärung aufgeführt und unter eigener Überschrift beschrieben:

-
- Refresh() nach cPickfield
- Damit alle beeinflussten Controls sofort uo to date sind
- „Picken" im Grid
- Wie binde ich Pickfield-Funktionalität in ein Grid ein?
- Datenformate und cUpdateSourceFields
- Keine Probleme mit Datenformatierung nach einem „Pick"
- Memofelder und Grids
- Wie zeige ich Memofelder im Grid an?
- Positionierung im Grid
- Wie kann ich bestimmen was oben oder unten steht?
- nFormStaus im Childform
- Anpassen der Formularreaktion von Parent/Child-Formularen

18.2. Refresh() nach cPickField

Eines der hilfreichsten Tools in VFX ist die cPickField-Klasse. Wenn Sie richtig angewendet wird, kann man sich eine Menge Arbeit sparen und macht ein Project 'more productive' .

Es gibt viele Dinge, die bei der Benutzung von cPickField bedacht werden müssen. Eines dieser Dinge ist die Nachbehandlung von Controls, die übernommene Daten anzeigen sollen:

Da VFX nach einem 'Pick' nicht weiß, ob es auf dem zugehörigen Formular Controls gibt die übernommene Daten anzeigen müssen, müssen Sie das selbst in die Hand nehmen. Die Daten sind übernommen, werden aber standardmäßig erst nach dem Speichern angezeigt. Zwei kleine Dinge müssen Sie dafür ändern:

1. Das Property ldovalid von unserem cPickFIeld muss auf .T. gesetzt werden, damit die vom PickField eigene valid() Methode aufgerufen wird. In dieser können Sie sich dann um die Controls kümmern. Im Powerbuilder für die cPickFieldKlasse können Sie ein entsprechendes Häkchen setzen, welches dann diese Eigenschaft automatisch setzt.

2. In dem valid vom txtField des cPickFields bauen Sie dann je einen refresh() pro Control ein, welches durch das „Picken“ beeinflusst wird.

VFX selbst macht keinen 'refresh()' aus gutem Grund: Nicht immer werden übernommene Daten angezeigt, und wo diese Daten gegebenenfalls angezeigt werden kann VFX auch nicht wissen. Das hätte zur Folge, dass VFX einen 'refresh()' immer auf die ganze Form machen müsste. - Und damit würde die Performance unserer Form ganz gehörig in den Keller gehen. Das wäre dann gar nicht mehr 'productive'.

18.3. „Picken“ im Grid

In einem Grid wird für die Suche und Übernahme von Daten aus anderen Tabellen nicht die cPickField-Klasse, sondern die cPickTextBox-Klasse benutzt.

Die Funktionalität ist natürlich ähnlich wie in cPickField, jedoch gibt es weder einen Knopf zum Aufruf der PickList noch ein Ergebnisfeld (txtDesc).

Um diese Klasse effektiv nutzen zu können, hat VFX auch hierfür einen Powerbuilder, der entsprechend dem Builder für die cPickField-Klasse funktioniert. Diesen können Sie allerdings nicht direkt über ein Kontextmenü aufrufen, da 1. Einstellungen für die Spalte vom Builder vorgenommen werden, und nicht für das Datencontrol in der Spalte, und 2. Ein Kontextmenü auf einem Grid aufgerufen sich immer auf das gesamte Grid und nicht auf die Spalte bezieht.

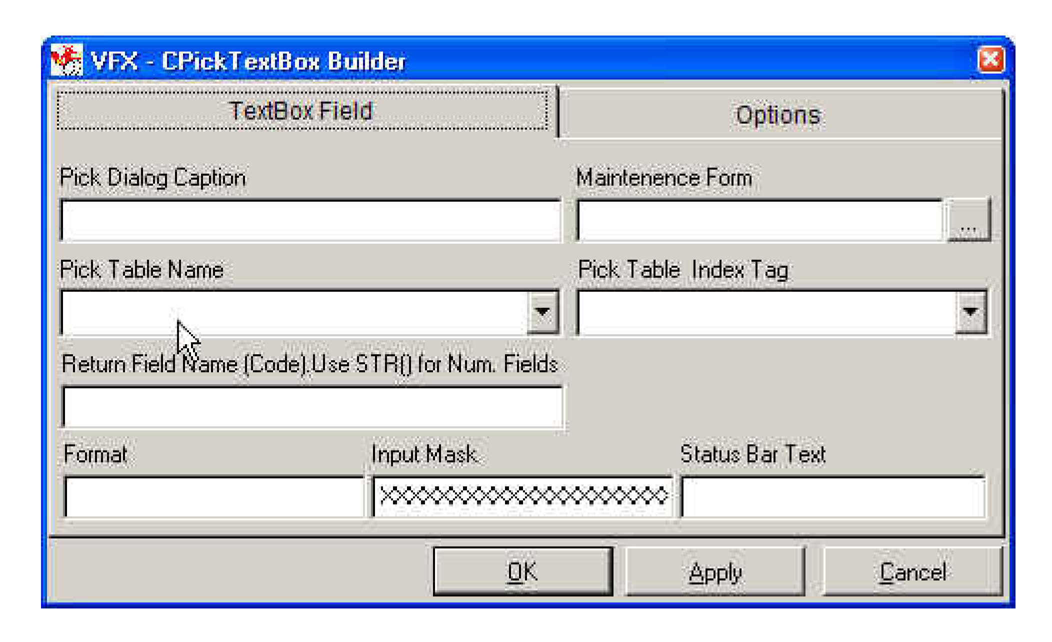
Nachfolgend der Weg zum Aufrufen des VFX-PowerBuilder:

- Wählen Sie die Spalte des Grids aus, in der die PickTextBox eingebaut ist.: *Wichtig: Die Spalte, nicht die PickTextBox selbst!*
- Jetzt aus dem Menü 'VFX' den Eintrag *VFX Powerbuilders* aufrufen.

VFX erkennt selbständig, ob das für uns passende Objekt den Fokus hat und ruft den entsprechenden Builder auf. Sollten Sie auf einem anderen Objekt stehen, wird automatisch der zugehörige VFX-Powerbuilder aufgerufen. Gibt

es keinen VFX-Powerbuilder, wird der interne VFP-Builder gestartet, sofern vorhanden.

In diesem Fall muss der VFX-Powerbuilder folgendermaßen aussehen:



In diesem Builder können Sie nun alle PickField-Einstellungen wie gewohnt vornehmen.

18.4. Datenformate und cUpdateSourceFields

In einem cPickField besteht die Möglichkeit weitere Datenfelder automatisch aus der Source-Tabelle zu befüllen. Dazu werden die Eigenschaften cUpdSourceFields und cUpdTargetFields benutzt, die auch im Builder schon gefüllt werden können.

Hierbei reicht es im Normalfall aus, jeweils eine Liste mit Tabellenfeldnamen getrennt mit Semikola einzugeben. Lediglich auf die Reihenfolge muss geachtet werden.

In Ausnahmefällen, muss allerdings noch Hand angelegt werden, wenn Datentypen nicht übereinstimmen. In der Sourcetabelle ist ein Datenfeld z.B. als Character definiert und in der Targettabelle soll dessen Inhalt in ein Feld gespeichert werden, welches als Numeric definiert ist.

Die 'Befüllung' der Targettabelle wird in der Methode updatetargetfields automatisch vorgenommen. Allerdings funktioniert dieses nicht, wenn die Datenformate nicht übereinstimmen, Entweder gibt es dann eine VFP-Fehlermeldung bez. der Datenformate oder, wenn man wie in diesem

Beispiel, einfach ein val() um das Feld in cUpdSourceFields bastelt, keine Fehlermeldung, aber es wird auch kein Wert übernommen.

Wenn man sich diese Funktion einmal ansieht, werden auch keine Überprüfungen mit Type() vorgenommen. Um trotzdem den Wert übernehmen zu können, müssen wir im Ereignis updatetargetfields unseres cPickFIeld folgendes tun:

```
=DoDefault()  && Ausführung des Klassencodes
Replace <TaregTable>.<Tagefield> with
val(<SourceTable>.<SourceField>) in <TargetTable>
```

Natürlich müssen wir dieses Feld aus cUpdTagerFields und cUpdSourceFields rausnehmen!

18.5. Memo-Felder und Grids

Wir benutzen oft, ja fast in jedem Formular die Klasse cGrid, um Auflistungen anzuzeigen. Hier, genauso wie in Standard-VFX-Formularen oder in der cPickList. In allen Grids ist es mit einem kleinen Trick möglich, Inhalte aus Memofeldern anzuzeigen:

Bei einem cGrid in einem Formular, z. B. die List-Seite, manipulieren Sie die Eigenschaft ControlSource z. B. wie folgt:

left(mytable.mymemofiled,20)

Das gleiche können Sie auch in der Eigenschaft cfieldlist einer cPIckTextBox oder eines cPickField machen. Einfach den 'normalen' Feldausdruck mit einem z. B. 'Left()' erweitern.

18.6. Positionierung im Grid

Jedes mal, wenn ein Sortierungsbefehl in einem cGrid gefeuert wird, bleibt der zuletzt ausgewählte Datensatz weiterhin ausgewählt. Man sieht es an der Markierung der aktuellen Gridline.

Um den Datensatzzeiger automatisch nach einer Neusortierung an eine andere Stelle des Grids zu stellen, muss man in jedem 'Header' des Grids eine Änderung vornehmen:

```
this.Parent.Parent.OnSetOrder(this.Parent)
GO top (zum Beispiel)
this.Parent.Parent.Refresh()
```

Natürlich wäre es einfacher die KLasse cGrid abzuleiten., wenn man nicht schon in allen Formularen das original cGrid einsetzen würde.

18.7. nFormstatus im Childform

In vielen Fällen ist es sinnvoll korrespondierende Daten zu Daten in einem Formular in einem zweiten Formular anzuzeigen. Normalerweise passiert das mit der onmore() Methode und einigen Eigenschaften der aufrufenden Formulares und der Childformulares.

Hier wollen wir aber Daten aus dem gleichen Datensatz in einem Child-Formular anzeigen, welches aber im EDIT-Modus geöffnet wird, wenn auch aufrufende Formul gerade im EDIT-Modus ist. Folgende Dinge sind zu tun:

1. Erstellen der Formulare (Parent und Child) wie gewöhnlich mit Aktivierung der entsprechenden Propertie-Boxen im VFX-Builder
2. Der Aufruf des Childformulares geschieht wie gewohnt mittels der onmore()-Methode
3. Im Childformular wird eine Eigenschaft erstellt, z. B.: 'nParentFormstate'
4. In der Load()-Methode des Child-Formulares wird diese Eigenschaft dann mit dem Namen des Parent-Formulares gefüllt:

   ```
   this.nPrentFormstate=_screen.activeform.nformstatus
   RETURN DODEFAULT()
   ```

5. Am Ende der Init()-Methode im Child-Formular wird dann der Status des Parent-Formulares abgefragt und der Status im Child-Formular entsprechend gesetzt:

   ```
   IF this.nParentFormstate=1
      this.onedit()
   ENDIF
   ```

Somit wird das Child-Formular nun im Edit-Modus geöffnet, wenn das Parent-Formular auch im Edit-Modus ist.

Für Fälle in denen Sie Generell den Status des aufrufenden Formulares übernehmen möchten codieren Sie anstatt des einfachen IF eine Casestruktur.

Beachten Sie bitte, dass der Formularstatus INSERT (2) nicht benutzt werden kann, wenn Sie im Childformular Daten des aufrufenden Forumlares benutzen möchten. In diesem Fall könnten Sie z. B. folgendermaßen vorgehen:

- Speichern Sie die Daten im aufrufenden Formular mit der Methode *onsave()*
- Legen Sie programmatisch einen Datensatz in der Tabelle für das Childformular an
- Wechseln Sie im aufrufenden Formular in den EDIT-Status (1)
- Rufen Sie dann das Childformular auf
-

Das ist sicherlich nur eine von mehreren Möglichkeiten die mir spontan einfallen, ich möchte Sie aber jetzt nicht alle im Detail wiedergeben, dazu wäre ein eigenes Themenpapier erforderlich.

18.8. Selbsterstellte Cursor

Normalerweise benutzen wir Tabellen oder Views in VFX-Formularen. Was aber nun, wenn wir Daten benutzen, die andernorts ihren Ursprung haben?

Wir erstellen uns einen Cursor. Damit der Cursor aber auch genauso funktioniert wie eine in der Datenumgebung geöffnete Tabelle, reicht es nicht aus eine Verbindung zur ControlSource-Eigenschaft herzustellen. Im load' des Formulars müssen wir dazu den Cursor als InitialSelected Alis in der Datenumgebung anmelden - ausser den Cursor zu erstellen natürlich:

```
CREATE CURSOR mycursor ...
this.dataenvironment.initialselectedalias="mycursor"
RETURN DODEFAULT()
```

Jetzt können wir den CURSOR wie eine Tabelle oder einen View benutzen.

19. Anhang 3 - Mehrsprachige Berichte mit VFX

Internationale Kunden bedienen: Professionell in der jeweiligen Landessprache.

Die Globalisierung lässt grüßen. Aber auch, wer sich 'nur' im europäischen Ausland bewegt, möchte seine Kunden in deren Landessprache anreden. Das schafft Vertrauen und stärkt die Geschäftsbeziehungen.

Als kleine Erweiterung der Reportgenerierung, mit wenigen Eingriffen des Entwicklers, könnt ihr für eure Kunden eine Erweiterung erstellen, die es dem Anwender ermöglicht einen Report in jeder gewünschten Sprache zu erstellen. Ihr braucht nicht zu wissen, welche Sprachen benutzt werden sollen, alles läuft über Tabellen, die der Anwender selbst pflegt, aber keine Änderungen an Reports vornehmen muss.

Hier findet ihr alles, was ihr zur Lösung benötigt. Umgesetzt mit Techniken von VFX

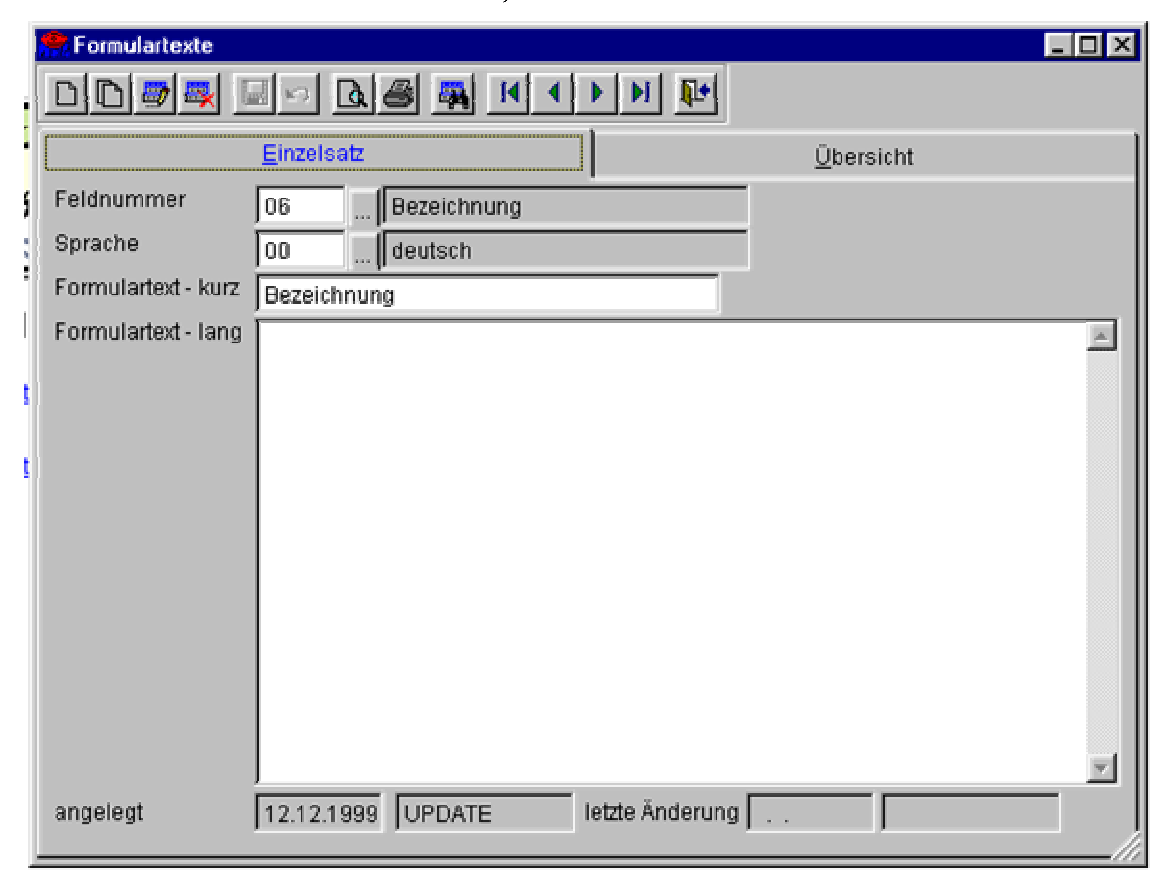
Dieser Lösungsansatz kann sehr gut als Erweiterung für das Projekt Runtime Localizationwith VFX benutzt werden!

1. Sie legen eine Tabelle an, in der u.a. Überschriften hinterlegt werden:

Field Name	Type	Width	Dec	Caption
CFELDNAME	C	2	0	Feldnummer
CFELDSPRACHE	C	2	0	Sprache
CFELDKURZ	C	40	0	Formulartext kurz
MFELDLANG	M	4	0	Formulartext lang
CINS_USR	C	10	0	angelegt von
DINS_DATE	D	8	0	angelegt am
CEDT_USR	C	10	0	geändert von
DEDT_DATE	D	8	0	geändert von
Record Length		84		

Tag Name	Primary	Key
FELDSPR	Yes	CFELDNAME+CFELDSPRACHE

2. Sie schreiben eine Funktion, die in der Tabelle nach einer Überschrift in der gewünschten Sprache sucht. Falls keine Überschrift gefunden wird, wird immer die deutsche genommen. Zu jeder Feldnummer kann es auch eine Langform geben, die immer genommen wird, wenn sie da ist. So werden nicht nur Überschriften übersetzt, sondern Sie können auch längere Texte in der gewünschten Sprache drucken. (z.B. "Keine Warenannahme in der Zeit vom 22.12.2001 bis 2.1.2002").
3. Sie erstellen ein Formular, in der der Anwender die Texte übersetzt:



Was in dem Feld "Feldnummer" zulässig ist, wird über eine zweite Tabelle festgelegt, die Sie entweder mit festen Inhalt ausliefern oder ebenfalls vom Anwender anpassen lassen können.

4. Im Report sind die Texte ebenso Textboxen wie auch die echten Daten. Als source rufen Sie hier die o. e. Funktion auf und übergeben die Gewünschte Sprache und die Feldnummer als Parameter.

```
txt_konstante(<cFeldnummer>,<cSprache>)
```

Sie können die Parameter entweder direkt angeben,

```
txt_konstante(„04“,“englisch“)
```

oder Sie übergeben die Werte implizit aus dem Druckaufruf:

```
txt_konstante(cFeldnummer,cSprache)
```

Die zweite Variante könnte bei entsprechender Definition der Tabellen die Verbindung mit der Sprachauswahl der Anwendung herstellen. So vervollständigen Sie die Automatik von VFX gridbasierte Reports der ausgewählten Sprache zu erstellen, mit der Möglichkeit das gleiche für Ihre Reports zu erreichen.

Im folgenden Beispiel sind Part information, Page, Part-No., Description und Price EUR auf diese Art enstanden:

Part information 20.12.1999 Page 1

Part-No.	Description	Price EUR
03 02 4530	Drive roller welded 730 long, RS 2 and 3 Track 2,20 and 2,35 m	1.602,00